Cristiano Lopes dos Santos Lopes

Licenciatura em Engenharia Informática

Aplicação Móvel para Visualização de Padrões Temporais

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática

Orientador : Prof. Doutor Nuno Manuel Robalo Correia, Prof. Associado com Agregação, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa

Júri:

Presidente: Doutor José Augusto Legatheaux Martins

Vogais: Doutora Maria Teresa Caeiro Chambel
Doutor Nuno Manuel Robalo Correia

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Março, 2012

Aplicação Móvel para Visualização de Padrões Temporais

Em memória do meu avô Carlos.

Agradecimentos

A realização desta dissertação não seria possível sem a ajuda prestada por um conjunto de pessoas e organizações.

Agradecer primeiramente ao Prof. Doutor Nuno Correia pela oportunidade que me ofereceu de trabalhar num projeto tão interessante como este. Agradecer também por toda a orientação técnica, constante disponibilidade e tranquilidade com que sempre abordámos as dificuldades que foram surgindo.

Agradecer a todos os elementos que participaram no projeto Time Machine[1], nomeadamente ao Samuel Del Bello, Marco Heleno, Sofia Oliveira, Jared Hawkey e Olivier Perriquet por toda a ajuda prestada. Ao Centro de Informática e Tecnologias de Informação (CITI/FCT) pelo apoio prestado através da bolsa de investigação que me atribuiu. À Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa e ao seu Departamento de Informática pelo fantástico ambiente de trabalho.

Agradecer aos meus irmãos, Henrique e Miguel. Ao Henrique pela paciência, boa disposição e companheirismo de todos os dias que me ajudaram a cada palavra escrita. Ao Miguel, pelos momentos passados juntos que me libertaram e inspiraram.

Aos meus pais, pelos valores que me ensinaram e amor que me deram. Ao meu pai pelos conselhos e motivação, à minha mãe pelo carinho e atenção.

À minha avó Maria José pelo apoio de todos os dias e aos meus avós maternos, José e Lia pela companhia e jantares às quintas.

À Tânia que me ilumina e inspira todos os dias e me ensinou que o entusiasmo é a força mais poderosa do Mundo. Também por ser a mais dedicada utilizadora da aplicação Time Machine.

À Patrícia por tudo. Ao André Ricardo pelos conselhos e ajuda prestada ao longo da dissertação e com quem foi um prazer trabalhar durante o meu percurso académico.

Grato a todos.

[1] PTDC/EAT-AVP/105384/2008

"Did you ever wake up to find,
A day that broke up your mind,
Destroyed your notion of circular time."
M. Jagger/K. Richards

Resumo

Na era da massificação do computador de bolso, torna-se cada vez mais interessante explorar a sua ubiquidade e capacidades sensoriais como forma de promover o auto-conhecimento e reflexão sobre a vida. Conhecer as características do movimento de um indivíduo permite extrair um conjunto de informação relevante no sentido de conhecer os padrões temporais e rotinas a que está sujeito. A utilização que um indivíduo faz do tempo e espaço é um indicador poderoso do seu modo de vida, estando associado aos eventos positivos e negativos que o marcam e a perceção dos ciclos que atravessa.

Para dar resposta a esta tarefa é necessário capturar de forma contínua a movimentação do utilizador ao longo do tempo e partindo dessa informação identificar os locais com significado para este. Através da utilização que o indivíduo faz destes locais, cumpre proceder à extração dos padrões temporais a que está sujeito e à identificação das suas rotinas e possíveis quebras. É também fundamental o desenvolvimento de formas de comunicação visual da informação extraída, que permitam cumprir os objetivos de reflexão e auto-conhecimento e possam constituir contributos artísticos com raízes no ato de viver.

A presente dissertação tem como objetivo a concretização do fluxo descrito, da captura à visualização, através do estudo, desenvolvimento, adaptação e integração de técnicas que a possibilitem.

Apresenta como resultado uma aplicação móvel para a plataforma Android e um conjunto de ferramentas aplicacionais que realizam os conceitos descritos através de um sistema informático, que captura e processa informação de localização, extrai conhecimento sobre rotinas e padrões temporais, e oferece um conjunto de interfaces visuais para estes.

Palavras-chave: Padrões de Tempo, Movimento Pessoal, Rotina Pessoal, Localização, Computação Ubíqua, Computação Móvel, Visualização

Abstract

With the dissemination of the pocket computer, it becomes increasingly interesting to explore its ubiquity and sensory capabilities as a way to promote self-awareness and reflection on life. Knowing the characteristics of an individual's movement can help to extract a set of relevant information in order to know the temporal patterns and routines to which he or she his subjected. The use of time and space by an individual is a powerful indicator of their lifestyle, associated with positive and negative events that remain as marks and the perception of the cycles someone goes through.

To meet this task it is necessary to track user's movement over time and based on this information to identify locations with meaning for that person. Based on the use that the individual does of this meaningful locations, it should proceed with the extraction of temporal patterns and identification of routines and possible breaks. It is also important to develop forms of visual communication of the extracted information, which can fulfil the goals of reflection and self-awareness and may constitute artistic contributions with roots in the act of living.

This dissertation aims at realising the described flow, from capture to visualisation, through the study, development, adaptation and integration of techniques that make it possible.

It presents as result a mobile application for the Android platform and a set of tools that implement the features described through a computer system that captures and processes the location information, to extract knowledge about routines and temporal patterns, and offers a set of visual interfaces to them.

Keywords: Time Patterns, Personal Movement, Personal Routine, Reality Mining, Location, Mobile Computing, Ubiquitous Computing, Visualization

Conteúdo

Lista de Figuras

Lista de Tabelas

Listagens

1

Introdução

Com o desenvolvimento e divulgação de dispositivos inteligentes de computação móvel, como *smartphones* e *tablets*, na rotina diária do ser humano, abre-se um mundo de possibilidades no estudo das características e dinâmicas do seu comportamento. Estes dispositivos são hoje parte integrante da vivência humana, acompanhando os indivíduos em praticamente todas as fases da sua vida e tornando-se verdadeiras extensões virtuais destes.

Dotados de capacidades sensoriais variadas como de localização, captura de imagem ou áudio, estes dispositivos podem registar informação contextual, expondo um conjunto de dados muito relevante para análise.

A comunicação da informação extraída a partir destes dados de volta aos indivíduos que a geraram, sugere o desenvolvimento de visualizações que promovam reflexão, oferecendo a possibilidade de um indivíduo ter uma visão exterior sobre a sua própria vida.

1.1 Motivação

A forma como um indivíduo se movimenta ao longo do tempo revela muito sobre a sua vida. Revela não só os sítios onde esteve, mas também os padrões temporais a que está sujeito, as rotinas que regem a sua vida. Como seres de hábitos, associamos os eventos especiais da nossa vida a quebras nestas rotinas, sejam elas definitivas ou temporárias e estejam associadas a sentimentos negativos ou positivos. A ocupação do espaço físico por cada indivíduo oferece por isso uma interessante possibilidade de auto-conhecimento, assim existam formas de captura, análise e transmissão da informação associada.

Estando a maioria dos dispositivos de computação móvel, especialmente *smartphones*, equipados com mecanismos de localização que permitem em todos os momentos saber a

sua posição geográfica, a materialização do conceito anteriormente descrito é agora uma possibilidade real e o principal fator motivacional da dissertação. Mais ainda quando estes dispositivos possuem uma adoção cada vez mais generalizada, em grande parte devido à existência de dispositivos com capacidades semelhantes em todos os segmentos de preço, cumprindo finalmente a promessa da computação ubíqua. Mas esta não abre apenas um mundo de possibilidades. Cria também nos utilizadores uma expetativa de *feedback* imediato que obriga a explorar os limites das capacidades de processamento e memória dos dispositivos móveis, motivando assim para uma adaptação de algoritmos e técnicas a este contexto.

Embora a recolha de dados de localização através destes dispositivos não seja propriamente uma novidade, serve usualmente o propósito de apresentação da posição do utilizador ao longo do tempo num mapa geográfico. Esta abordagem caracteriza-se normalmente por um processamento dos dados pouco elaborado que conduz a uma diminuta extração de informação com relevância para o utilizador. Também a visualização em mapa geográfico, apesar de muito efetiva em certos domínios de aplicação, não é o veículo mais interessante para a reflexão e promoção do auto-conhecimento. Estas lacunas motivam o estudo e aplicação de técnicas mais consentâneas com as possibilidades oferecidas pela informação disponível.

1.2 Descrição e Contexto

O projeto Time Machine tem por objetivo a apresentação dos padrões temporais a que um indivíduo esta sujeito no seu dia a dia, através da recolha e análise de informação acerca dos seus movimentos. Na era da computação ubíqua, pretende tirar partido da omnipresença dos dispositivos de computação móvel nas nossas vidas, como forma de promover a reflexão e interrogação sobre a nossa utilização do tempo e do espaço. Expondo a rotina e suas quebras através de visualizações pessoais e subjetivas, pretende construir uma cartografia pessoal do utilizador que assuma um significado único para este. Focado no indivíduo, pretende promover um meio de auto-conhecimento que não lhe é proporcionado enquanto está ocupado a viver.

O projeto Time Machine pretende também apresentar um vislumbre do futuro do indivíduo, suscitando a interrogação sobre a influência do passado na sua definição, expondo ao mesmo tempo as limitações e potencialidades dos algoritmos de inteligência artificial e da computação ubíqua em geral.

O desenvolvimento duma aplicação móvel, situada na categoria de arte multimédia, é o meio escolhido para a materialização do conceito do projeto. Numa altura em que a palavra aplicação, aplicada ao contexto de software, se tornou numa *buzzword* de conhecimento generalizado, faz todo o sentido utilizar este meio como proposta de reflexão e auto-conhecimento sobre o ato de viver. Integrando elementos de áreas distintas do conhecimento como a ciência, design e arte, o projeto pretende refletir estas sensibilidades no produto final, uma peça de software acessível ao maior número de pessoas possível.

A dissertação enquadra-se no desenvolvimento do projeto descrito, uma colaboração entre o Centro de Investigação em Informática e Tecnologias de Informação (CITI[1]) da Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa (FCT/UNL[2]) e o coletivo artístico CADA[3], sendo financiado pela Fundação para a Ciência e Tecnologia (FCT/MEC[4]). A dissertação surge da necessidade de estudar, desenvolver e integrar técnicas que permitam materializar o conceito nuclear do projeto através do desenvolvimento da aplicação móvel e dos seus diversos componentes, desde a captura de dados de movimento ao processamento e extração de informação e visualização.

1.3 Solução Apresentada

A solução apresentada no sentido de materializar o conceito subjacente ao projeto Time Machine passa, como referido anteriormente, pelo desenvolvimento de uma aplicação móvel. A aplicação desenvolvida captura de forma constante a localização do utilizador, recolhendo informação acerca da forma como este se movimenta, através de dois sistemas de localização: GPS e localização via rede. Em tempo real, a aplicação deteta os locais onde o utilizador permanece durante um tempo assinalável e vai construindo uma base de conhecimento acerca dos locais com significado para o utilizador, segundo critérios de frequência de acesso e duração de estadia. Esta base de conhecimento evolui de forma dinâmica, e depois de uma aprendizagem inicial, a aplicação começa a percecionar os eventos que ocorrem da mesma forma que o utilizador: um novo local é novo para ambos, um local conhecido é conhecido por ambos, ambos conseguem definir a relevância de um local.

A aplicação monitoriza a utilização destes locais com significado de modo a extrair os padrões temporais a que o utilizador está sujeito, identificando assim rotinas e possíveis quebras. Este processo é também dinâmico, estando o sistema atento à tipologia dos eventos que vão surgindo e que podem ser de natureza mais ou menos definitiva, reajustando a sua aprendizagem de acordo.

Um conjunto de visualizações fazem também parte da aplicação desenvolvida, permitindo uma visão sobre os dados que pretende estimular reflexão e um olhar exterior sobre o dia a dia do utilizador. As visualizações desenvolvidas funcionam de forma dinâmica, refletindo o contexto atual através de um olhar imediato sobre os eventos que vão ocorrendo, tendo em conta o conhecimento obtido pela aplicação no passado. Uma experiência de previsão da atividade do utilizador foi incluída numa destas visualizações indicando um dos caminhos possíveis na abordagem a este problema.

A plataforma utilizada no desenvolvimento da aplicação foi a Android, escolhida pelo seu carácter de código aberto, abrangência de utilização, disponibilidade de ferramentas de desenvolvimento e facilidade de distribuição. É uma plataforma móvel de

[1] citi.di.fct.unl.pt
[2] www.fct.unl.pt
[3] www.cada1.net
[4] www.fct.pt

nova geração com tendência de crescimento que permite o desenvolvimento de aplicações móveis com bom desempenho, suportada por um grande conjunto de dispositivos dos mais variados segmentos de mercado.

A solução desenvolvida inclui ainda uma aplicação *desktop* multi-plataforma que possibilita a reprodução do processo de aprendizagem a partir de um conjunto de dados capturado anteriormente, desenvolvida como ferramenta de suporte às experiências de processamento e extração de informação implementadas na aplicação móvel. Um conjunto de protótipos *desktop* de visualização são também parte integrante da solução desenvolvida, formando pequenas experiências visuais que se espera possam vir a ser adaptadas e integradas na aplicação móvel ou simplesmente servir de inspiração a futuros desenvolvimentos.

1.4 Principais Contribuições

As principais contribuições que resultam da solução desenvolvida são listadas de seguida.

- Uma aplicação móvel que captura informação sobre a localização do utilizador, identificando os locais com significado para este, e que extrai os padrões temporais a que esta sujeito.
- Definição e implementação de um algoritmo de identificação de pontos de estadia em tempo real.
- Definição e implementação de um algoritmo de identificação de locais com significado para o utilizador em tempo real.
- Modelação de padrões temporais do utilizador através da definição de modelo de utilização do tempo e transições entre estes.
- Um conjunto de visualizações contextuais que promovem reflexão e o auto-conhecimento do utilizador em relação aos seus padrões temporais em ambiente de computação móvel.
- Uma aplicação *desktop* que permite a simulação da execução dos algoritmos desenvolvidos para funcionarem em tempo real sobre um conjunto de dados de localização previamente capturado.
- Um conjunto de protótipos *desktop* de visualização a serem adaptadas e integradas na aplicação móvel.

No contexto do projeto Time Machine a aplicação móvel e um conjunto de ferramentas computacionais foram aceites para exibição e demonstração na conferência AmI 11-International Joint Conference on Ambient Intelligence 2011[5]. A exibição gerou bastante

[5] http://www.ami-11.org

curiosidade por parte dos participantes que se revelaram interessados pela ideia e sua execução, originando proveitosas discussões sobre o conceito e dificuldades de implementação inerentes. Foi sem dúvida um ponto de confirmação da relevância do projeto, tanto na vertente conceptual como técnica.

1.5 Estrutura do Documento

O presente documento encontra-se estruturado segundo seis capítulos, apresentados na listagem seguinte.

Capítulo 1 - Introdução O capítulo inicial apresenta uma visão geral da dissertação através da descrição da sua motivação e temática referindo o contexto em que se insere. Termina descrevendo a solução desenvolvida e enumerando as principais contribuições resultantes.

Capítulo 2 - Trabalho Relacionado Este capítulo descreve e discute o estado da arte relevante para a persecução dos objetivos da dissertação. Efetua este levantamento segundo os principais blocos que compõe a aplicação: Captura e Processamento de Informação Geográfica, Extração de Conhecimento e Visualização.

Capítulo 3 - Visualização de Padrões Temporais: Ferramentas e Aplicação Móvel Este capítulo apresenta as aplicações e protótipos desenvolvidos sob o ponto de vista da funcionalidade desenvolvida. Inicia com a descrição do Simulador, seguido dos protótipos de visualização e termina com a aplicação móvel.

Capítulo 4 - Realização do Sistema Este capítulo versa os aspetos mais relevantes da realização das diversas aplicações desenvolvidas. Foca-se maioritariamente sobre a implementação da aplicação móvel, mencionando também os aspetos principais respeitantes ao Simulador.

Capítulo 5 - Avaliação Este capítulo efetua uma avaliação do trabalho desenvolvido, partindo do *feedback* obtido através da realização de inquéritos a utilizadores.

Capítulo 6 - Conclusões e Trabalho Futuro O capítulo final efetua uma análise crítica sobre todo o trabalho realizado no âmbito da dissertação e reflete sobre possíveis futuros desenvolvimentos a efetuar no contexto em que esta se insere.

2

Trabalho Relacionado

O presente capítulo pretende informar acerca de um conjunto de trabalhos relacionados com o âmbito da dissertação nas diversas áreas abordadas ao longo do seu desenvolvimento. Apresenta uma descrição de técnicas e metodologias que constituem a base da solução desenvolvida ou que de alguma forma contribuíram para uma tomada de decisão informada. A estrutura apresentada acompanha as fases de desenvolvimento, começando pela Captura e Processamento de Informação Geográfica na secção 2.1, seguida de uma secção relacionada com Extração de Conhecimento em 2.2 terminando com a apresentação de trabalho relacionado com a área da Visualização em 2.3. Na secção 2.4 apresentam-se algumas considerações finais acerca das técnicas e metodologias apresentadas.

2.1 Captura e Processamento de Informação Geográfica

A captura e processamento de informação geográfica formam a base de suporte para o desenvolvimento do trabalho de visualização e interação em dispositivos móveis que se pretende executar. A captura desta informação e sua associação a uma linha temporal, formam o conjunto de dados a partir dos quais se pretende fazer emergir os padrões temporais. Nesta secção serão introduzidos sistemas de localização, técnicas de processamento e sistemas de armazenamento passíveis de utilização em ambiente móvel.

2.1.1 Captura de Informação Geográfica

Grande parte dos dispositivos de computação móvel atuais estão dotados de mecanismos de captura de informação geográfica que lhes permite a obtenção da localização dos seus

utilizadores de forma eficaz. De seguida, efetua-se uma análise de sistemas de localização considerados relevantes para o objetivo da dissertação e um estudo da implementação dos mesmos na plataforma Android.

2.1.1.1 Sistemas de localização

Os sistemas de localização permitem a captura de informação relativa ao posicionamento de um dispositivo de computação móvel. A utilização destes sistemas como ferramenta de recolha dos dados que permitirão a extração de informação e criação de modelos de interação e visualização sobre os mesmos, obriga a um levantamento das opções mais relevantes para o cumprimento dos requisitos pretendidos. Para que se possa proceder a uma análise das várias opções, é importante definir uma taxonomia que permita classificar os sistemas de localização. A taxonomia escolhida para proceder à classificação é apresentada em [HB01] e indica oito categorias das quais se destacam aquelas que têm maior preponderância como fatores de escolha para dar resposta ao problema a ser estudado.

Física ou Simbólica O sistema de localização pode ser classificado como fornecedor de localização física ou simbólica. Uma localização física corresponde a uma medida concreta, como um par latitude, longitude, que permite situar a medida num sistema de referência concreto. Uma localização simbólica está mais relacionada com uma medida abstrata, uma ideia da posição relacionada com a natureza do objeto a localizar. Usualmente as localizações físicas podem ser estendidas com algum simbolismo, através do cruzamento de informação, e esse é um dos objetivos da presente dissertação.

Absoluta ou Relativa Um sistema de localização absoluta utiliza um sistema de referência comum a todas as suas medições. Por oposição, um sistema de localização relativa utiliza uma referência própria para cada entidade posicionada.

Exatidão e Precisão A exatidão e precisão estão relacionadas com a medida de qualidade da localização indicada pelo sistema de posicionamento. A exatidão define o detalhe das posições indicadas e é expressa tipicamente numa medida de distância. A precisão é um valor indicativo da confiança em obter o valor de exatidão indicado, sendo expressa numa percentagem.

Escala Define essencialmente o domínio de abrangência do sistema de localização dada a estrutura que o suporta. Usualmente é possível, se o sistema for escalável, aumentar a escala através de um aumento da capacidade da infraestrutura.

A aplicação que se pretende desenvolver indica a utilização de um sistema de localização que ofereça características de **localização física, absoluta e escala global**. Quanto à precisão e exatidão, dado que se pretende identificar os locais por onde o utilizador se movimenta e permanece, é fundamental apontar para o máximo de precisão e exatidão

que se possa atingir. Existe adicionalmente a restrição do sistema de localização escolhido ser suportado, ou passível de desenvolvimento, na plataforma Android e o mais abrangente possível neste domínio dada a heterogeneidade de dispositivos suportados por este sistema.

Apresenta-se de seguida um levantamento de sistemas de localização que vão de encontro à maioria das características anteriormente enunciadas.

2.1.1.2 GPS

O sistema de localização denominado Global Positioning System, abreviadamente GPS, é uma criação do Departamento de Defesa Americano, e teve na sua génese motivações militares, no sentido de providenciar um serviço de localização espacial e temporal, disponível em qualquer parte da Terra onde exista uma linha desimpedida entre o dispositivo cliente e pelo menos quatro satélites que façam parte do sistema (para a precisão anunciada). O seu cariz exclusivamente militar foi progressivamente abandonado e hoje em dia é utilizado também com fins comerciais, em sectores como a navegação para aviação, navegação automóvel e mesmo para posicionamento de dispositivos móveis, como sejam *smarthphones* ou *tablets*. O sistema tornou-se totalmente operacional durante o ano de 1994 e tem sofrido sucessivas melhorias, que permitiram um incremento na precisão e exatidão do mesmo, e até uma queda no preço dos recetores e consequente massificação da sua presença em dispositivos de computação móvel de todos os segmentos de mercado.

Funcionamento O funcionamento do sistema GPS assenta na conjugação de 3 estruturas de suporte: Espacial, Controlo e Recetora.

Estrutura Espacial A Estrutura Espacial é composta por um conjunto de 31 satélites, divididos em grupos partilhando a mesma órbita e compreendendo tantas órbitas como grupos. Desta forma é garantido que na grande maioria dos locais da Terra é possível obter uma linha desimpedida que permita obter sinal emitido por, pelo menos, 6 satélites diferentes. Cada um destes satélites emite um sinal numa determinada frequência, descodificado como uma mensagem contendo o tempo de emissão, informação sobre a órbita e um conjunto de informações sobre todos os outros satélites.

Estrutura de Controlo A estrutura de controlo é constituída por um conjunto de estações base terrestres utilizadas para permitir correção de erros na estrutura espacial.

Estrutura Recetora Constituída pelo conjunto de todos os recetores, independentes e desconhecidos das restantes estruturas, que descodificam o sinal emitido pela estrutura espacial, procedendo de seguida ao cálculo do posicionamento com base nos dados contidos no sinal descodificado.

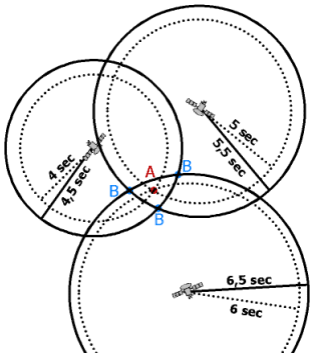


Figura 2.1: Cálculo de posição GPS: triangulação baseada no tempo de propagação do sinal

Cálculo da Posição Em [Lam05] são apresentados os 5 passos executados por um dispositivo recetor com vista a efetivar o cálculo da sua posição através do sistema GPS.

1. O recetor captura o sinal enviado por um satélite pertencente à estrutura espacial.

2. Depois de descodificar o sinal, determina a diferença entre o tempo corrente medido pelo recetor e a estampilha temporal extraída do sinal capturado.

3. Sabendo que o sinal emitido viaja à velocidade da luz, calcula a distância a que se encontra o satélite emissor.

4. Repete o processo para pelo menos mais 2 satélites, mas usualmente para um total de pelo menos 4, para correção de erros introduzidos por fatores que interfiram com o sinal.

5. Com base nas 3 medidas referentes a posições conhecidas, executa um processo de triangulação da distância obtendo assim a sua posição como pode ser observado na figura 2.1.

Fontes de Erro A principal fonte de erro no cálculo da posição neste sistema tem a ver com a existência de obstáculos ao sinal emitido, que o obriguem a despender mais tempo para chegar ao recetor, em relação ao caso ideal de uma linha de vista direta e desimpedida. Este efeito designado *multipath*, é mais grave em zonas de grande densidade habitacional, em que muitos obstáculos se interpõem entre o recetor e a propagação do sinal. O atraso introduzido por este efeito, leva a um enviesamento nas distâncias calculadas pelo recetor dado o tempo de propagação desviado do tempo de propagação em linha direta. A qualidade do recetor, nomeadamente do relógio disponível, pode também influenciar a exatidão do posicionamento. Apesar de ser possível efetuar alguma correção do desvio do relógio utilizando a informação conjunta de vários satélites, um relógio com uma deriva muito grande conduzirá naturalmente a maiores erros no posicionamento.

2.1.1.3 Localização baseada em redes celulares

A localização baseada em redes celulares, aproveita a infraestrutura existente, que suporta usualmente serviços de comunicações de operadoras móveis, para providenciar estimativas mais ou menos precisas da localização dos dispositivos. É bastante interessante fundamentalmente pela ubiquidade da infraestrutura de suporte. Existem várias técnicas de localização utilizadas para recolher informação geográfica com base na infraestrutura de rede celular. Seguidamente apresentam-se as que têm revelado maior utilização prática.

A-GPS Surgiu como forma de mitigar alguns dos problemas de localização do GPS base. Essencialmente, permite dotar o posicionamento GPS de possibilidade de localização dentro de edifícios e diminuir de forma significativa o tempo necessário para a obtenção da primeira localização (*first fix*). É considerado um sistema de localização híbrido, pelo facto de utilizar a infraestrutura de rede celular, mas também o posicionamento via satélite (GPS). O problema do *time to first fix* é resolvido inquirindo a rede celular sobre informação do posicionamento dos satélites e suas órbitas o que leva a uma descoberta célere dos mesmos e dos dados que transmitem. Adicionalmente, a rede pode ainda informar o dispositivo acerca da estampilha temporal associada a um processo de posicionamento, por possuir relógios com menos deriva, o que leva a um incremento na qualidade das medidas obtidas. A infraestrutura pode mesmo calcular a posição com base em medidas GPS recolhidas na torre mais próxima, permitindo assim aos dispositivos terem uma noção aproximada da sua posição mesmo em locais onde não consiga captar sinal (geralmente dentro de edifícios).

Esta técnica encontra-se presente na generalidade dos dispositivos móveis de nova geração, permitindo ganhos não apenas a nível de exatidão, precisão e cobertura, mas também em eficiência energética, visto que minora o problema do *first fix* que é a fase que requer maior dispêndio de energia na localização por GPS. O facto de não existir um esforço extra por parte do dispositivo, visto que o mesmo já tem impreterivelmente de comunicar com a infraestrutura celular pelo decorrer da sua função de comunicação, permite poupanças energéticas apreciáveis, à custa de algum tráfego de dados.

Cell-Id Consiste simplesmente em obter a localização da torre de emissão de sinal, correspondente à célula da rede a que o dispositivo se encontra ligado. Uma célula GSM pode variar entre 2 e 20km de diâmetro em relação à cobertura que oferece. Dois melhoramentos são usualmente utilizados, aumentando bastante a exatidão dos resultados. A deteção do sector intracelular em que o dispositivo se encontra e o cálculo do tempo de propagação do sinal entre o dispositivo e e a antena emissora de sinal. Este último permite estimar a distância que separa dispositivo e torre emissora para a célula de uma forma semelhante ao que acontece no GPS. Além disso, a utilização de três ou mais células e dos tempos de propagação respetivos permite a aplicação de técnicas de multilateração conduzindo a resultados de melhor qualidade.

É de notar que em dispositivos móveis orientados a telecomunicações, este tipo de localização não impõe, para além do necessário tempo de processamento, qualquer fardo adicional a nível energético, dado que todas as ligações já têm de ser obrigatoriamente mantidas dada a natureza dos dispositivos.

2.1.1.4 Localização via Wi-Fi

Partindo da ideia apresentada pelo sistema RADAR [BP00] de utilizar pontos de acesso Wi-Fi para localizar indivíduos dentro de edifícios, é apresentado em [RMI07] um sistema de *tracking* de localização de utilizadores denominado LifeTag. O sistema consiste num serviço web que estima a localização do utilizador à partir dos endereços MAC, e força de sinal associada, capturados por um dispositivo sensor. Este sensor consiste num telemóvel ou num dispositivo especialmente desenvolvido para o efeito, que procura em cada momento informação sobre todos os pontos de acesso Wi-Fi alcançáveis. Esta informação é de seguida enviada para o serviço web que possui uma base de dados que relaciona a informação dos pontos de acesso com coordenadas geográficas. A base de dados é capturada *offline*, através de levantamentos dos pontos de acesso existentes e da sua associação manual com as coordenadas geográficas. O serviço web dado o conjunto de pontos de acesso enviados pelo sensor, inicia uma busca na base de dados devolvendo como localização a posição correspondente ao ponto de acesso com maior força de sinal.

Este sistema de localização pode à partida parecer apenas utilizável em situações de pequena escala. Os próprios autores posicionam-no como um método de localização dentro de edifícios, no entanto com o aparecimento de bases de dados de grande escala contendo informação geográfica (latitude, longitude) associada a pontos de acesso Wi-Fi como a SkyHook [1], pode providenciar um sistema de localização de propriedades muito interessantes. O nível de exatidão e precisão pode até ser superior ao encontrado em soluções baseadas em redes celulares, dado o raio de alcance de um ponto de acesso Wi-Fi ser bastante limitado.

A nível energético existe um pequeno custo associado, embora de nível inferior ao imposto pelo sistema GPS. Como ponto negativo, o facto de regiões menos desenvolvidas ou não urbanas não possuírem pontos de acesso que possam ser utilizados para cálculo de localização torna inútil o sistema. Apesar desse facto, de uma perspetiva de integração com um sistema alternativo como o GPS, permitiria uma poupança de energia considerável em zonas densamente urbanas com o GPS a servir de recurso em regiões fora deste domínio.

2.1.1.5 Arquitetura de suporte a serviços de localização na plataforma Android

A plataforma Android oferece suporte para serviços de localização, através da sua API, desde a versão 1.0. É portanto transversal a todos os dispositivos que se baseiam neste

[1] http://www.skyhookwireless.com/

sistema independentemente da versão adotada, desde que exista no dispositivo o necessário hardware de suporte conforme o sistema de localização pretendido.

O modelo utilizado para fornecer acesso aos dados geográficos baseia-se no padrão de desenho Observer sendo o registo para notificação efetuado explicitamente através da classe *LocationManager*[2]. Esta é acedida, não através de instanciação direta, mas através do padrão de desenho Singleton, que devolve uma instância da classe sobre a qual é possível requerer notificações para mudanças de posição, ou fazer pedidos como a última localização conhecida. A primeira opção é indicada para situações em que é necessário efetuar o *tracking* da posição ao longo do tempo, por oposição à segunda que é adequada a aplicações que necessitam obter informação posicional apenas em situações esporádicas, ou em consequência de uma ação ou pedido do utilizador. É exemplo da primeira opção, uma aplicação para oferecer indicações de condução (*turn-by-turn*) em tempo real e da segunda uma aplicação para sugerir restaurantes próximos da localização atual do dispositivo.

***Tracking* de Localização** Para dar resposta à necessidade de seguir de forma contínua a localização do dispositivo, é necessário registar a aplicação como Observer do LocationManager. Para tal é invocado o método `requestLocationUpdates` com a assinatura indicada em 2.1 sobre a instância de LocationManager anteriormente obtida. Esta é a melhor arquitetura para dar resposta aos requisitos, pois não exige recursos adicionais com o crescimento do número de observadores. O LocationManager pode ser dispensado de tentar obter localização se não existe nenhum observador e consome os mesmos recursos independentemente do número de aplicações no papel de observadoras. Contudo o observado consome sempre os recursos necessários para o observador registado com os parâmetros mais exigentes (intervalo de tempo e distância mais curtos). Também na plataforma IOS uma arquitetura muito semelhante é utilizada [3] .

Listing 2.1: Android: método para requerer notificações de localização

```
1 public void requestLocationUpdates(String provider,
2                     long minTime,
3                     float minDistance,
4                     LocationListener listener)
```

Tabela 2.1: Android: parâmetros método requestLocationUpdates

Nome	Tipo	Descrição
provider	String	Sistema de Localização a Utilizar (passive, network, gps).
minTime	long	Intervalo temporal de referência para novas notificações.
minDistance	float	Intervalo de distância mínimo para novas notificações (m).
listener	LocationListener	Referência para objeto de *callback*.

[2] http://developer.android.com/guide/topics/location/obtaining-user-location.html

[3] https://developer.apple.com/library/ios/navigation

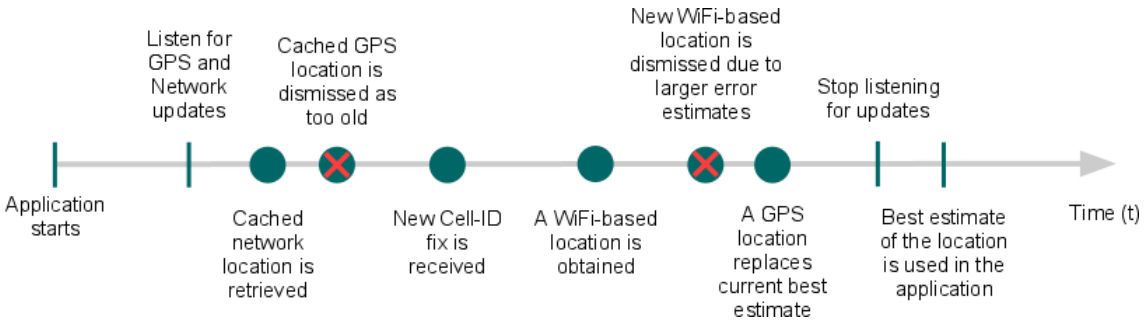


Figura 2.2: Fluxo para *tracking* de localização.

É de particular interesse discutir os argumentos *minTime* e *minDistance*, visto que são apresentados na API como altamente relacionados com o desempenho energético do dispositivo, fator de grande relevância no desenvolvimento de uma aplicação com objetivos de *tracking* de localização que não leve ao abandono dos utilizadores devido à incomportável degradação da autonomia do dispositivo.

A plataforma não impõe qualquer limite aos valores utilizados como argumentos para o método `requestLocationUpdates`. A decisão é deixada ao critério da aplicação que o invoca, sendo que o valor de *minTime* é utilizado apenas como um indicativo e não como algo a ser criteriosamente cumprido. Serve apenas o propósito de otimizar a poupança de energia. A API desaconselha firmemente a utilização de valores abaixo de 60000 ms, o que pode levar a uma degradação total da autonomia do dispositivo. Já o valor de *minDistance* determina a variação mínima entre localizações detetadas, reduzindo o fardo de processamento introduzido pela notificação mas não está diretamente relacionado com a poupança de energia respeitante ao dispendioso processo de obtenção de localização por parte do hardware GPS. A utilização de valores 0 para ambos os argumentos garante a notificação de novas localizações o mais frequentemente possível.

É possível pedir ainda notificação de localizações baseado em perfis de energia ou de precisão pretendidos, o que adiciona uma abstração interessante, que torna mais fácil a adaptação às necessidades das aplicações. É recomendado um fluxo aplicacional na API para aplicações de *tracking* de localização, como se pode observar na figura 2.2. Como se pode verificar depois do processo de registo como Observer a aplicação vai sendo notificada ao longo do tempo, devendo de acordo com as seus objetivos atualizar a melhor estimativa (baseada na exatidão) ou simplesmente guardar ou utilizar todas as medidas de que é notificada. Quando não existir mais necessidade de recolher informação, deverá proceder à remoção da lista de Observadores, para que os sistemas de localização sejam desligados (caso em que mais nenhuma aplicação está registada) ou menos processamento seja necessário na notificação, o que conduzirá a uma poupança energética.

Obtenção de Localização Atual Quando apenas é necessária a obtenção esporádica ou relacionada com algum evento proveniente de interação com o utilizador de localização, não é necessário registar a aplicação para ser notificada de mudanças de posição

bastando obter uma instância para LocationManager, invocando de seguida o método apresentado em 2.2.

Listing 2.2: Android: método para obter última localização

```
1 public Location getLastKnownLocation(String provider)
```

Tabela 2.2: Android: parâmetros método getLastKnownLocation

Nome	Tipo	Descrição
provider	String	Sistema de Localização a Utilizar (passive, network, gps).

O método devolve a última localização recolhida, para o fornecedor indicado em argumento, podendo esta informação estar desatualizada ou não. Cabe à aplicação aferir a qualidade da medida devolvida, em correspondência com as necessidades da aplicação. O resultado pode ser meramente um valor mantido em cache que esteja desatualizado por exemplo por o dispositivo ter sido desligado e movido para uma localização diferente, ou nenhuma aplicação ter requerido recentemente notificação de localizações. Se tal facto suceder e a medida devolvida for inexistente ou não satisfizer os requisitos pretendidos pela aplicação a mesma deverá iniciar o processo indicado em 2.1.1.5, obter uma localização fresca de forma assíncrona e logo de seguida remover-se da lista de Observers podendo assim o sistema de localização ser desligado acautelando as questões energéticas.

Dados Recolhidos Os dados recolhidos são entregues às aplicações através de objetos da classe *Location*, contendo informação da medida e também da qualidade associada. A lista dos principais valores recolhidos pode ser encontrada na tabela 2.3 bem como uma breve explicação do conteúdo de cada um deles. Os sistemas de localização que podem

Tabela 2.3: Android: dados entregues pelo sistema de localização

Nome	Tipo	Descrição
Accuracy	double	Precisão da medida em metros.
Altitude	double	A altitude em relação ao nível do mar.
Latitude	double	Longitude.
Longitude	double	Longitude.
Provider	String	Sistema de Localização que originou a medida (passive, network, gps).
Time	Datetime	Tempo UTC associado à medida.

estar na origem da medição podem ser restringidos aquando do registo para obtenção de localizações, ou da obtenção da última medida conhecida, ou utilizados em simultâneo, e têm uma relação direta com os sistemas descritos em 2.1.1.1.

gps Tal como descrito em 2.1.1.2 e 2.1.1.3 utiliza o sistema GPS, recorrendo à rede para providenciar um *time to first fix* menor. Consome um nível de energia acima em relação aos restante fornecedores.

network Um sistema híbrido, baseado na rede celular como descrito em 2.1.1.3 combinado com localização de pontos de acesso Wi-Fi num sistema que segue os mesmos princípios descritos no sistema RADAR (2.1.1.4) mas com uma base de dados de grande escala.

passive Fornecedor que dispensa o registo explícito para notificação de localizações. Se outras aplicações estiverem registadas para receber notificações, as notificações geradas para essas serão entregues também às registadas como passivas.

A utilização dos vários sistemas de localização em simultâneo é permitida e garante a maior cobertura possível, embora nem sempre mantendo o mesmo grau de precisão e exatidão. De referir que para a utilização de qualquer um destes sistemas por parte da aplicação, o utilizador deve ativa-los através das preferências gerais de sistema. A aplicação não pode de forma independente proceder à sua ativação, estando limitada a reencaminhar o utilizador para as preferências gerais de sistema.

2.1.2 Processamento de Informação Geográfica

A natureza dos dados de localização descrita anteriormente e a previsível escala inerente aos requisitos a que se pretende dar resposta, obrigam à utilização de técnicas de processamento com vista a garantir robustez aos erros introduzidos no processo de captura e também capacidade de lidar com grandes quantidades de dados em ambiente móvel. A necessidade de deteção da presença do utilizador num determinado local e do cálculo do tempo de estadia no mesmo, indicam um caminho que não pode ser necessariamente orientado a uma lógica de simples registo coordenada a coordenada da sua posição. Independentemente do sistema de localização utilizado na captura, existe sempre um nível de ruído associado, quer por erro introduzido pela natureza do sistema de localização, quer pela irrelevância, de certos pontos capturados na ótica do seu significado para o utilizador.

Em [dB11], é apresentado um algoritmo de filtragem de dados geográficos que com uma passagem sobre os dados segundo uma ordenação temporal crescente (mais antigo primeiro), extrai um conjunto de pontos de estadia (Staypoints). De uma só vez este algoritmo permite sintetizar o conjunto de dados e eliminar medições que não possuam qualidade suficiente (medida em precisão).

O algoritmo utiliza dois parâmetros, tempo e distância, para proceder à criação dos Staypoints. Sempre que a distância entre a coordenada i e a coordenada j é maior que um certo limite ($distThresh$) e desde que a precisão de j seja aceitável, é efetuado um cálculo do tempo decorrido entre a captura da coordenada i e da coordenada j. Se o tempo decorrido for maior que um limite estipulado ($timeThresh$), um novo Staypoint é criado e adicionado ao resultado com a coordenada média de todas as coordenadas capturadas entre i e j, tempo de início correspondente à captura da coordenada i e tempo de fim correspondente à captura da coordenada j. Caso este último teste temporal falhe, os dados são descartados, tomando i o valor de j e iniciando-se novo processo até que não

existam mais dados. O algoritmo que implementa este comportamento é apresentado em 2.3.

Este algoritmo possui propriedades interessantes, dado que pode ser adaptado para efetuar uma filtragem à medida que os dados vão sendo capturados (por oposição a uma aproximação *batch*) em tempo real e é resistente ao ruído, tanto o introduzido pelo sistema de localização utilizado na captura, como o relacionado com a ausência de significado relevante da medição para o utilizador por ter sido um mero local de passagem.

Listing 2.3: Algoritmo de extração de StayPoints

```
i = 0, pointNum = |p|;
startI = 0, endI;
while i < pointNum do
  j = i + 1;
  endI = i
  while j < pointNum do
    dist = Distance(pi, pj);
    if dist > distThresh && pj. precision > precisionMin then
      deltaT=pj. T - pi. T
      if  deltaT > timetThresh then
        if SP. isNotEmpty; then
          SP. last. Trajectory = {pk|startI <= k <= startJ};
        end if
        S. coord = ComputMeanCoord({pk|i <= k <= j});
        S. arvT = pi. T;
        S. levT = pj. T;
        SP. insert (S);
        startI = j;
      end if
    i = j;
    break;
    end if
  j = j + 1;
  end while
end while
return SP;
```

A definição dos limites utilizados como parâmetros do algoritmo assume grande relevância nos resultados obtidos, carecendo assim de um estudo e adaptação aos requisitos aplicacionais e também a expectável heterogeneidade na qualidade dos dispositivos de captura. O autor sugere um tempo de estadia de 3 minutos e uma distância de 3 metros como valores que permitem resultados apreciáveis sem perda de informação.

Um algoritmo baseado nos mesmos princípios, mas com duas ligeiras variações, é apresentado em [MGP10]. Este não leva em conta a qualidade das medições efetuadas, através da sua precisão, mas introduz um novo parâmetro, o tempo máximo entre medições. Este tempo máximo impede que seja criado um Staypoint entre duas medições se tiver passado um período de tempo muito grande. Permite evitar que eventos como a

perda de sinal do sistema de localização ou a indisponibilidade do mesmo num largo período de tempo causem um enviesamento no processamento. É por isso mais resistente a casos de ausência de dados, situação previsível em ambientes de computação móvel, com recursos energéticos limitados.

2.1.3 Armazenamento de Informação Geográfica

O armazenamento de informação geográfica, numa perspetiva de *tracking* das localizações importantes para o utilizador, num dispositivo móvel, apresenta desafios que derivam da quantidade de dados gerados, mas também da multidimensionalidade e estrutura dos mesmos. Os algoritmos de processamento e extração de informação, bem como a componente de visualização, envolvem a análise da base de conhecimento construída pela aplicação, sendo por isso fundamental que a mesma seja servida de forma ágil e eficiente. Adicionalmente o requisito de que a aplicação deve funcionar assente num modelo *standalone*, obriga ao estudo de sistemas de armazenamento de informação que sejam exclusivamente contidos no dispositivo móvel, com as já conhecidas limitações de memória e processamento. Apresenta-se de seguida um levantamento e análise de dois sistemas de bases de dados relacionais que funcionam segundo um modelo *standalone* e que podem ser utilizados na plataforma Android.

2.1.3.1 SQLite

A biblioteca SQLite apresenta-se como uma implementação de um motor de base de dados relacional SQL, que dispensa servidor, requer pouca configuração e é totalmente independente (embutida)[Res]. É especialmente desenhada a pensar em ambientes com pouca memória disponível, embora o seu desempenho melhore se as restrições neste domínio forem reduzidas. Garante todas as propriedades ACID e possui as características ideais para armazenamento de dados com alguma estrutura em ambientes de mobilidade e restrições de conectividade, memória e processamento. Implementa a quase totalidade da norma SQL92 com exceção de algumas funcionalidades, nomeadamente : Right Outer Join, Full Outer Join e vistas que não suportam operações de escrita.

Uma interessante extensão ao SQLite, que pode ser incluída na biblioteca através da compilação com a modificação de uma *flag*, é a capacidade para criação de índices r-tree. São especialmente úteis para otimizar consultas multidimensionais como no caso de consultas geográficas, geralmente baseadas em relações de proximidade entre locais definidos por coordenadas geográficas (latitude e longitude).

O modelo relacional oficialmente suportado pela plataforma Android utiliza este sistema de gestão de base de dados, oferecendo uma API para a execução das operações usuais. No entanto nem sempre a versão oferecida é a mais atualizada e muitas das extensões não estão disponíveis, sendo de especial importância referir que a extensão geográfica não é oferecida, não permitindo assim a utilização de índices r-tree. Não existe também atualmente nenhuma implementação da API oferecida na plataforma Android

para J2SE o que impede o desenvolvimento de código totalmente compatível fora do ambiente móvel.

2.1.3.2 H2

O sistema de base de dados H2 [Mue11] consiste numa biblioteca de código aberto totalmente escrita em Java, que implementa a API JDBC para acesso a bases de dados relacionais. Implementa a totalidade da especificação SQL e apresenta modos de compatibilidade com diversos SGBD como o PostgreSQL e o MySQL. Permite vários modos de operação, com especial relevância para o modo *standalone* em que permite executar uma conexão a uma base de dados baseada num ficheiro local.

Não possui suporte para índices multi-dimensionais mas é possível através de uma ferramenta fornecida converter cada chave multidimensional para um valor escalar através de técnicas de bit-interleaving. Depois da conversão pode ser utilizado um índice b-tree para o qual existe suporte nativo do sistema. Na documentação são indicados ganhos de desempenho num fator de 5 sobre a utilização de apenas um índice na primeira coluna da chave multidimensional pretendida.

A implementação oferecida permite de forma fácil desenvolver uma aplicação em J2SE e portar o seu código diretamente para a plataforma Android, dada a utilização da especificação JDBC oferecida em ambas as plataformas de desenvolvimento. No entanto o suporte a esta especificação na plataforma Android não é oficial o que não garante uma continuidade da sua presença em novas iterações da plataforma. Esta situação esta relacionada com o facto da API JDBC obrigar ao carregamento de um *driver* pela aplicação que a pretenda utilizar. Assim cada aplicação desenvolvida na plataforma deveria incluir o seu próprio *driver* o que não beneficiaria a plataforma em termos de consistência e interligação entre aplicações. O consórcio responsável, definiu que seria incluído apenas um SGBD, com API desenvolvida de raiz, garantido consistência e desempenho máximo no armazenamento relacional em toda a plataforma.

A nível de performance são anunciadas algumas desvantagens em relação à implementação SQLite oficial, principalmente em operações de conexão com a base de dados e de escrita.

2.1.3.3 Geohash

Como anteriormente discutido, não é usual encontrar SGBD assentes num modelo *standalone* que ofereçam a possibilidade de utilização de índices r-tree, críticos para a execução de perguntas do tipo geográfica, que formam a base do processamento e extração de conhecimento que se pretende seja efetuado pela aplicação. Além disso, as características inerentes a dispositivos de computação móvel, sugerem um uso cuidado da memória e capacidades de processamento disponíveis pelo que é conveniente que se trabalhe apenas sobre um subconjunto dos dados exclusivamente necessários ao processo a executar.

O sistema Geohash, desenvolvido como parte da solução para a criação do serviço

web http://geohash.org, é uma estrutura de dados espaciais hierárquica, que divide o espaço em regiões dispostas em grelha. O objetivo do projeto http://geohash.org é a criação de URLs que representem uma localização geográfica, mas apresentou como resultado adicional um sistema de codificação de coordenadas geográficas com propriedades que refletem a noção de proximidade na sua codificação e permite a definição arbitrária da precisão pretendida, para além de resumir um par de coordenadas numéricas (latitude, longitude) numa cadeia de caracteres [con].

Funcionamento Para obter o *hash* correspondente a um par de coordenadas geográficas, é estipulado um máximo de bits utilizados para a representação dos valores. O processo de transformação consiste no aproveitamento do facto de os valores de latitude e longitude se situarem exclusivamente no intervalo -90 a 90 e -180 a 180 respetivamente. Através da divisão desse intervalo no ponto médio, é verificado se o valor se situa no intervalo inferior ou superior e atribuído o valor do bit correspondente (0 inferior, 1 superior). A atribuição é feita à posição i correspondente ao número de divisões efetuadas até ao momento. De seguida é atualizado o limite inferior ou superior do intervalo conforme o bit atribuído. No final, o conjunto de bits é transformado para base 32. De notar que os bits em posições ímpares são usados para representar as latitudes e os pares as longitudes.

O processo de descodificação é inverso ao de codificação com os valores finais de latitude e longitude a serem obtidos através do ponto médio do último intervalo calculado.

A precisão é determinada pelo comprimento do *geohash* utilizado como apresentado na tabela 2.4.

Tabela 2.4: Precisão sistema Geohash [con]

Comprimento Hash	Erro latitude	Erro longitude	Erro Km
1	± 23	± 23	± 2500
2	± 2.8	± 5.6	± 630
3	± 0.70	± 0.7	± 78
4	± 0.087	± 0.18	± 20
5	± 0.022	± 0.022	± 2.4
6	± 0.0027	± 0.0055	± 0.61
7	± 0.00068	± 0.00068	± 0.076
8	± 0.000085	± 0.00017	± 0.019

Limitações A principal limitação deste sistema de representação, reside no facto de a propriedade de um prefixo comum para localizações geograficamente próximas, não se verificar em situações de comparação entre dois locais muito próximos mas situados em lados opostos do equador ou diferentes meridianos. Para ultrapassar este facto, em pesquisas por proximidade, devem ser analisadas não apenas um par de codificações mas também o conjunto de codificações vizinhas (áreas) de um deles.

Aplicabilidade Este sistema pode ser muito útil à aplicação dado que permite a transformação do par de atributos (latitude,longitude) num só atributo sobre o qual é possível a criação de índices em qualquer dos sistemas de bases de dados anteriormente enunciados. Definindo a precisão certa é possível efetuar perguntas que excluam à partida elementos geograficamente distantes do que se pretende analisar, pagando apenas um pequeno custo de armazenamento de informação redundante ao par (latitude,longitude), mas poupando memória, processamento e tempo de execução.

2.2 Extração de Conhecimento

A extração de conhecimento a partir de dados de localização assume importante relevância na tarefa de modelação da rotina do utilizador. Na presente secção, efetua-se um levantamento de trabalho relacionado com esta área, que possibilite o conhecimento de técnicas relevantes para a construção de um sistema capaz de responder adequadamente aos requisitos.

2.2.1 Extracção de Locais com Significado

Kang et al. definem um local com significado como :

> ...a locale that is important to an individual user and carries important semantic meanings such as being a place where one works, live, plays, meets socially with others, etc.[KWSB05]

A extração de locais com significado para o utilizador forma a base principal na descoberta das suas rotinas pessoais. O conhecimento destes locais e o seu cruzamento com a variável tempo, são a melhor informação disponível para a execução da tarefa de modelação da sua rotina. Igualmente importante é a pouca predisposição por parte dos utilizadores, para participar neste processo de descoberta de conhecimento. Tal como referido em [PM09], um processo manual ou semiautomático de classificação de locais, pode tornar-se irritante e distrativo, levando ao abandono da aplicação. Este facto indica um caminho que deverá ser essencialmente não-supervisionado e automático, que torne o processo de extração de informação o mais invisível possível, pelo que se restringe este estudo a métodos que privilegiem estas características nomeadamente a métodos de *clustering*.

Destacam-se duas abordagens para endereçar o problema de extração de locais com significado, uma denominada *density-based* e outra denominada *grid-based*.

Na dissertação "Extracção de Informação de Padrões Pessoais de Movimento" [dB11] é apresentado um método *density-based*, baseado no algoritmo DJ-Cluster [ZZF+04] com uma ligeira variação de forma a permitir contemplar não só a dimensão frequência mas também a dimensão tempo despendido no local.

O algoritmo recebe como entrada um conjunto de Staypoints e calcula a vizinhança de cada um. A vizinhança é constituída por todos os pontos que distem menos que um valor

Eps incluindo o próprio ponto. Caso a cardinalidade da vizinhança seja maior ou igual ao número mínimo de pontos *minPts*, um novo local importante é criado. Caso algum dos pontos pertencentes à vizinhança já faça parte de algum local importante, então as vizinhanças são fundidas num só local. Por outro lado um local com significado pode ser criado se uma vizinhança somar entre si uma duração de estadia mínima *MinDrt*. O algoritmo é apresentado em 2.4.

Listing 2.4: Algoritmo para extração de locais importantes: Density-Clustering

```
while there is at least one unprocessed point p in samples do
  Compute the density based neighborhood N(p) with Eps and MinPts.
  if N(p) is null (p is not in a cluster) then
    Label p as noise.
  else
    if N(p) is density joinable to an existing cluster then
      Merge N(p) and all its density joinable clusters.
    else
      Create a new cluster C based on N(p).
    end if
  end if
end while
```

O algoritmo DJCluster modificado apresenta propriedades interessantes, nomeadamente a capacidade de formação de *clusters* de formato não circular, sendo garantido que todos os locais importantes contêm pelo menos um ponto e que todos os pontos ou pertencem a um grupo ou são ruído. Adicionalmente, considerando as suas características iterativas, pode ser adaptável para dar resposta a um requisito de processamento em tempo-real.

Como pontos negativos a apontar a este algoritmo será consensual dizer que os limites utilizados no decorrer da sua aplicação influenciam o comportamento do mesmo, podendo os parâmetros utilizados ser satisfatórios para um utilizador com um certo comportamento e não satisfatórios para outro inserido numa realidade distinta.

O facto de algoritmos *density-based* não limitarem o tamanho máximo de um local com significado é alvo de crítica em [MGP10] e [ZZXY10]. Existe a possibilidade real de um local crescer indefinidamente, principalmente se o utilizador se movimentar de forma lenta numa determinada região, estendendo assim sucessivamente a sua fronteira pela adição de Staypoints na periferia.

Em [ZZXY10] é apresentada uma alternativa *grid-based* descrita de seguida. É feita uma divisão de um mapa em células de formato quadrado, com tamanho do lado igual a $d/3$ que levará a que nenhum dos locais importantes extraídos tenha dimensão superior a $d \times d$. Para cada célula são atribuídos todos os Staypoints de entrada que nela estejam contidos. Terminado este passo, é iniciado o processo de *clustering* que consiste no varrimento das células por ordem de número máximo de Staypoints contidos. Para cada uma destas células é criado um local importante, através da união da célula atual com todas as células na sua vizinhança. Todos os Staypoints pertencentes à união destas células são atribuídos ao novo local importante. O processo repete-se enquanto existirem Staypoints

que não pertençam a nenhum local importante. Uma descrição mais formal do algoritmo pode ser encontrada em 2.5.

Listing 2.5: Algoritmo para extração de locais com significado: Grid

```
Input : d (max size of a Location), List<StayPoints> StayPoints
Output : List<Location> output

List<GridCell> g = gridDivision(d);
for(GridCell gi : g){
  gi.sp = getStayPointsThatBelongTo(gi);
}
for(StayPoint sp : StayPoints){
  sp.location = null;
}
while(isThereAStayPointWithoutLocation(Staypoints)){
  GridCell current =  getCellWithMaxNumberOfStayPoints(g);
  List<GridCell> neighbours = getNeighbours(current,g);

    Location newloc = new Location(getCentroidCoordinate(current,neighbours));
  output.add(newloc);

  assignNewLocToStayPoints(current,neighbours);

  g.remove(current);
  g.removeAll(neighbours);
}
```

A dificuldade na aplicação deste algoritmo, prende-se essencialmente com o contexto da aplicação que se pretende desenvolver. O mesmo é apresentado assumindo uma região delimitada a partir da qual é possível executar facilmente a divisão em células de tamanho $d/3$ e forma estática para cada região. Tendo em conta a natureza bastante imprevisível dos locais denominados importantes para o utilizador e o conhecimento geral acerca da forma como este se movimenta, utilizar como único critério um valor d de entrada que determina toda a partição do espaço parece ser demasiado restritivo. Existe ainda o problema da escala global que se pretende obter ou seja como dividir um espaço imprevisível nas respetivas células. Uma possível solução seria a utilização da técnica de GeoHash (2.1.3.3) para executar a divisão do espaço em células, utilizando como valor para d uma das precisões possíveis.

2.2.2 Modelação da Rotina do Utilizador

A construção de um modelo que represente a rotina do utilizador é essencial para permitir a representação dos padrões temporais a que este está sujeito. A utilização desta abstração permite também expressar de forma mais consistente os desvios da rotina e ilustra-los visualmente. Como referido em [FGP08] a tarefa de inferir e modelar a rotina de seres humanos através de dados recolhidos por dispositivos móveis é bastante

complexa principalmente devido ao ruído presente nos dados capturados e a descontinuidades na captura (dados em falta). Basear a construção deste modelo na recolha apenas de dados de localização não foge a regra, tanto mais que vários problemas podem ocorrer desde erros introduzidos pelo processo de captura, até ao simples facto da indisponibilidade do dispositivo móvel de captura por falta de bateria.

Em "What Did You Do Today? Discovering Daily Routines from Large-Scale Mobile Data" [FGP08] é apresentada uma metodologia para extração de rotinas pessoais a partir de um conjunto de dados de localização para cem pessoas. Inicialmente uma simplificação é introduzida na metodologia apresentada. Cada local capturado para uma pessoa é automaticamente catalogado como sendo casa, trabalho, outro ou sem receção a partir dum conhecimento prévio dos dados das torres celulares utilizadas para a obtenção da localização. Depois de executada esta etiquetagem, o dia é dividido em parcelas de trinta minutos e a cada parcela é atribuída a etiqueta única correspondente ao local onde se permaneceu mais tempo. O próximo passo do processo consiste na divisão do dia em parcelas correspondentes a períodos do dia pré-determinados (0-7am, 7-9am, 9-11am, 11am-2pm, 2-5pm, 5-7pm, 7-9pm, and 9-12pm), para que o processo seja resistente ao ruído e a variações na rotina consideradas irrelevantes para as características que os autores pretendem extrair. Se "Sair de Casa pela manhã" é um valor que se pretende extrair dos dados é irrelevante sair às 7h30am ou as 8h00am, o que dentro do contexto de rotina simplificada é um processamento interessante. Todo este processo visa a utilização de algoritmos de processamento de tópicos a partir de conjuntos de palavras, pelo que de seguida todas as parcelas de trinta minutos são processadas uma a uma de forma a serem construídas *location words*. Uma *location word* consiste em três etiquetas consecutivas (1h30m) anexadas ao período da divisão maior a que a parcela de trinta minutos pertence. Obtido este conjunto de palavras, algoritmos de processamento de tópicos a partir de conjuntos de palavras são utilizados visando a criação de um modelo probabilístico que permite inferir os tópicos e palavras mais frequentes para cada período do dia.

Este caso é paradigmático da dificuldade da tarefa em questão. O conjunto de pessoas utilizadas no estudo partilham o mesmo local de trabalho (professores e alunos no MIT), é introduzida a simplificação do conhecimento prévio das etiquetas associadas a cada local e a rotina de um indivíduo é reduzida à relação entre casa, trabalho e outros. Se é uma assunção razoável que a maioria das rotinas pessoais formem a sua base na relação casa-trabalho ou casa-casa, não o é o facto de assumir uma tipologia de trabalho homogénea para todas elas e menos ainda considerar que todos os locais para além de casa e trabalho são de importância idêntica.

No âmbito do projeto TimeMachine o documento "TimeMachine - Final Report" [Per11] foi desenvolvido com a finalidade de apresentar um modelo teórico para a representação da rotina de um indivíduo. Inicialmente são indicados os pressupostos e requisitos subjacentes ao modelo desenvolvido :

- São considerados os ciclos de tempo fundamentais que advêm de normas culturais,

físicas e biológicas nomeadamente dia, semana e mês.

- O sistema deve responder em tempo-real ou muito aproximado, aprendendo com o conhecimento passado e reagindo de acordo.
- Um período de treino e de confirmação é assumido para todos os cálculos permitindo que o sistema esteja informado antes de atuar.
- Dois locais merecem tratamento especial, Casa e Trabalho. O primeiro é caracterizado como o local onde um indivíduo passa mais tempo e onde dorme, o segundo como o segundo local onde o utilizador passa mais tempo.

O modelo preconizado assenta no conceito de *time containers* descritos como modelos de utilização do tempo (rotinas, regularidades, locais mais importantes). O sistema decide se o utilizador deixou de estar num determinado *time container* transitando para um novo e guardando o anterior com base nos perfis de dia. Perfis de dia são descritos como conjuntos de medidas estatísticas agrupadas segundo diferentes períodos de tempo.

Este modelo teórico permite pela sua orientação à diferença da norma, modelar o conceito de rotina pessoal. No entanto existe a dificuldade de detetar as mudanças de modelo de uso do tempo o que será fundamental para que os valores estatísticos recolhidos tenham significado. Se valores de um modelo novo forem agregados a valores de um outro modelo em uso, as medidas serão apenas um conjunto de valores sem grande significado, com os valores de um modelo a cancelarem os do outro. A definição arbitrária destes eventos pode fazer com que o modelo perca generalidade dado que os eventos corretos para uma pessoa podem não o ser para outra. Além disso a escolha dos atributos a recolher é relevante e difícil, levantando problemas como a representação estatística da regularidade de um local onde se costuma ir duas vezes no mesmo dia. Este é um caso de atributo que segue uma distribuição multimodal e sobre o qual não podem ser aplicadas diretamente técnicas estatísticas como a média e o desvio padrão.

2.2.3 Predição de Comportamento do Utilizador

A predição do comportamento do utilizador é também um dos objetivos da dissertação. Em trabalhos que tentam executar este tipo de extração de informação são usualmente apresentados modelos estatísticos de previsão do próximo local para onde o utilizador se deslocará. Andrei Papliatseyeu e Oscar Mayora em [PM09] efetuam um levantamento de algumas técnicas entre as quais se destaca a denominada Cadeia de Markov. Uma Cadeia de Markov pode ser descrita como um grafo dirigido em que os nós representam estados e os arcos probabilidades de transição para outros nós. Pode assim ser definida uma matriz de probabilidades de transição de um estado para outro. Este é um sistema sem memória, em que a transição para um estado depende apenas do conhecimento do estado atual. Uma extensão a este modelo probabilístico, Cadeias de Markov de ordem m, pode ser construído como uma extensão ao modelo base de forma a considerar uma

memória de tamanho m. Neste caso a transição do estado atual para um futuro depende não só do estado atual mas também da trajetória de dimensão m percorrida até à chegada ao estado atual. Esta aproximação é mais exigente a nível de cálculo computacional exigido e necessita também de uma base de conhecimento maior, caso contrário a inexistência de conhecimento passado conduzirá à redução das probabilidades de transição entre estados a zero.

Andrei Papliatseyeu e Oscar Mayora ainda em [PM09] criticam as técnicas anteriores pelo facto de apenas se preocuparem com o próximo local onde é previsto que o utilizador esteja. Fatores como o tempo de estadia e a regularidade na utilização de um espaço são ignorados, assim como o contexto que pode ser sugerido pelo tempo atual (e.g. ,"Hoje é sábado e estou em casa, para onde vou¿' por oposição a "Estou em casa, para onde vou¿').

Olivier Perriquet [Per11] baseado no modelo referido anteriormente (2.2.2), apresenta algumas formas de predição que vão de encontro às críticas apresentadas anteriormente. Tendo as agregações estatísticas por perfil, sugere a utilização destes valores para prever o comportamento geral do indivíduo. Dado o dia atual e o conhecimento recolhido para os diversos perfis é possível designar o comportamento estatístico observado como o expectável para o futuro. Uma falha nesta previsão é vista como um sinal de comportamento desviante da rotina.

2.3 Visualização

Edward Tufte, na sua obra "Envisioning information" [Tuf90], descreve o problema da visualização de informação, como a tarefa hercúlea de escapar à *flatland*. *Flatland*, é apresentada como uma metáfora para o objetivo perseguido pela disciplina de visualização de informação, de representar um Mundo multidimensional, em dispositivos limitados a apenas duas dimensões. Tufte, ainda em [Tuf90], introduz cinco conceitos de grande importância para a visualização de informação:

Micro/Macro Readings A facilidade de leitura de uma representação não advém da simplificação da mesma, mas do contexto adequadamente estruturado de informação complexa e detalhada. Para clarificar é necessário adicionar detalhe. A visualização de informação complexa e multidimensional deve ser providenciada de forma a que o observador consiga obter uma visão geral da mesma, que providencie contexto, mas também a possibilidade de clarificar detalhes e precisar informação, sem sair do contexto. Este tipo de estrutura permite a comparação da visão global com a de detalhe facilitando uma compreensão geral dos dados.

Layering and Separation A separação e estruturação das visualizações permite eliminar a confusão e impossibilidade de leitura sobre dados de natureza complexa. A combinação de representações numa só permite ao mesmo tempo o cruzamento e a separação de informação desde que efetuado de forma harmoniosa. A utilização

da cor, textura ou contraste permite definir camadas de informação sobre as quais é possível raciocinar de forma global mas correlacionada com as várias dimensões. A definição das camadas não é trivial, podendo surgir o efeito de $1 + 1 = 3$ se as combinações provocarem um efeito não previsto quando visualizadas em conjunto.

Small Multiples A utilização da repetição como método para ilustrar a diferença ou a alternativa. É um meio eficaz de mostrar variações subtis nos dados, pela repetição dos objetos representados, transformados por variações nos dados de suporte.

Color and Information A cor é apresentada como uma ajuda importante na tarefa de escapar da *flatland*, permitindo o endereçamento da multidimensionalidade dos dados, ao ser utilizada como forma de catalogamento, medição, representação e decoração. A cor pode assim ser utilizada como meio eficaz para representar nomes, quantidades, relações, representações e como componente estética.

Narratives of Space and Time As dimensões tempo e espaço são geralmente bastante difíceis de representar. A conjugação destas duas variáveis permite a representação de movimento e fluxo de informação ao longo do tempo e do espaço. A utilização de visualizações dinâmicas, com o tempo representado como o controlador da evolução das outras dimensões, pode ser a resposta para um problema complexo de mapeamento de dados multidimensionais em representações de natureza bidimensional. Usualmente sempre que existem dados relacionados com a dimensão tempo a mesma tende a ser menosprezada em relação às restantes dimensões [AAD+10], pelo que é necessário desde o início da definição da visualização tomá-la em consideração.

2.3.1 Visualização de Padrões Temporais

A informação temporal/espacial é um exemplo ilustrativo das dificuldades em escapar à *Flatland* descrita por Tufte. A sua multidimensionalidade, usual complexidade e elevada quantidade de dados obriga à aplicação de técnicas como as descritas em 2.3. Esta necessidade aumenta em ambientes de computação móvel, em que o tamanho do ecrã disponível para interação visual com o utilizador é menor e existem algumas restrições impostas pela capacidade de processamento.

Em [AAG03] é apresentada uma visualização denominada Space-Time Cube que utiliza a dimensão vertical para representar o tempo, projetando as coordenadas geográficas nas restantes dimensões. Utiliza ainda cores e tamanho dos círculos para representar os valores respeitantes a outras dimensões dos dados. Um exemplo deste tipo de visualização pode ser observado em 2.3. É uma visualização bastante efetiva que permite representar conceitos como a proximidade ao tempo presente, através da altura dos pontos. Possui ainda a particularidade de permitir que o utilizador manipule a visualização e tome vários pontos de observação o que lhe permite diferentes perspetivas dos mesmos dados.

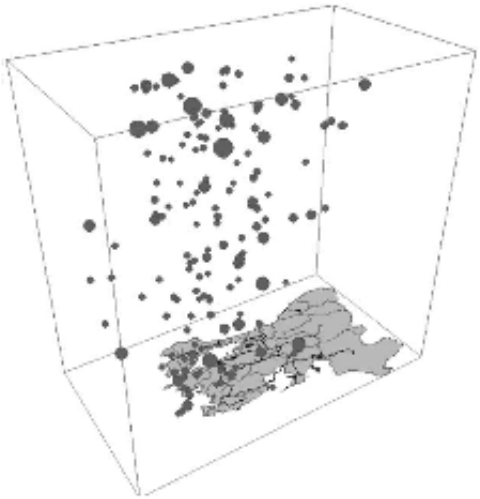


Figura 2.3: Space-Time Cube

Vários métodos de representação do tempo têm sido utilizados podendo ser definida a seguinte classificação [AAD+10] :

- Linear ou cíclica.
- Intervalos de tempo ou pontos no tempo.
- Perspetiva múltipla, ordem cronológica ou ramificações de tempo.

A utilização de visualizações cíclicas permite mostrar de forma mais clara padrões que se repetem ao longo de um intervalo de tempo, o que as torna muito interessantes para informação sobre rotinas pessoais de movimento. Já a utilização de representações lineares do tempo é mais apropriada para visualizar um histórico de dados, concentrando a atenção num período especifico e na relação direta tempo/dados.

Usualmente as representações que lidam com uma componente temporal utilizam pontos no tempo, mostrando uma evolução linear das restantes dimensões. Por vezes é interessante assentar a representação em intervalos de tempo, fixos ou dinâmicos, que permitam uma comparação entre períodos temporais e uma macro leitura dos dados. Isto é alcançável essencialmente por duas vias, tratamento dos dados através de operações de síntese ou visualizações manipuláveis que permitam uma análise integrada de várias escalas. Um exemplo da primeira opção pode ser vista na figura 2.4, onde uma agregação das trajetórias ao longo dos dias é efetuada sobre os dados, sendo posteriormente utilizada a saturação da cor como representação da densidade das trajetórias.

Muito frequente é a utilização de representações do tempo ordenadas cronologicamente, por ser a forma mais natural de raciocinar sobre esta dimensão. No entanto os métodos que permitam visualizar o tempo através de várias perspetivas ou ramificações, são bastante efetivos na apresentação de previsões de valores futuros sujeitos a diversas opções ou condicionalismos.

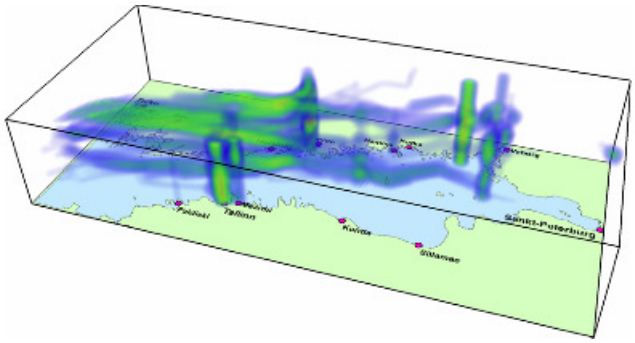


Figura 2.4: Visualização de trajetórias de petroleiros com agregação por densidade

Uma visualização mais conservadora, assente numa representação linear, ordem cronológica e com base numa escala diária é apresentada em [KK06]. Apesar das suas características não deixa de ser uma visualização muito efetiva, que permite claramente formar uma ideia sobre as atividades desenvolvidas por um utilizador ao longo de um dia, sendo possível observar vários dias em simultâneo e assim compará-los. Prevê ainda a possibilidade de filtrar por tipo de atividades o que confere um modo de micro-leitura ao nível da atividade ou subconjunto pretendido, por oposição à macro-leitura da visualização global. Um exemplo pode ser observado na figura 2.5, onde é ainda possível verificar a utilização da côr como característica da atividade o que confere uma ideia de ritmo ao longo e dentro dos dias.

Este tipo de visualização pode ser essencialmente visto como um *chromogram* tal como descrito em [WVH07]. Neste trabalho, palavras resultantes de edições de artigos na Wikipedia são transformadas para cores utilizando as primeiras três letras de cada palavra. A primeira letra é utilizada como valor para *hue* a segunda para *saturation* e a terceira para *brightness* com estes dois últimos limitados a um intervalo de forma ao *hue* ser percetível. Apesar deste método de mapeamento ser completamente cego no que diz respeito à semântica das palavras, é muito eficiente na dispersão das palavras no modelo de cor e é determinístico, o que permite comparação entre a visualização para o artigo antes e depois de ser editado. Sempre que se pretenda desenvolver visualizações que permitam comparação, é importante a utilização de métodos determinísticos nas heurísticas de mapeamento entre variáveis do domínio dos dados e do domínio visual.

Em [SlM08], a tentativa de revelar o padrão de trabalho ao longo do tempo para um conjunto de utilizadores resulta numa visualização (figura 2.6) mais uma vez baseada na intensidade de cores que consegue transmitir claramente o ciclo de trabalho esperado (segunda a sexta). O eixo vertical representa os dias da semana, começando de baixo para cima no Domingo e sendo a cor mais próxima do vermelho indicativa de maior atividade laboral.

Uma aproximação interessante ao problema da integração das escalas de tempo surge em [VW07] em que um círculo é utilizado para representar as imagens capturadas por um sistema de vídeo vigilância, com a parte exterior a representar as imagens capturadas

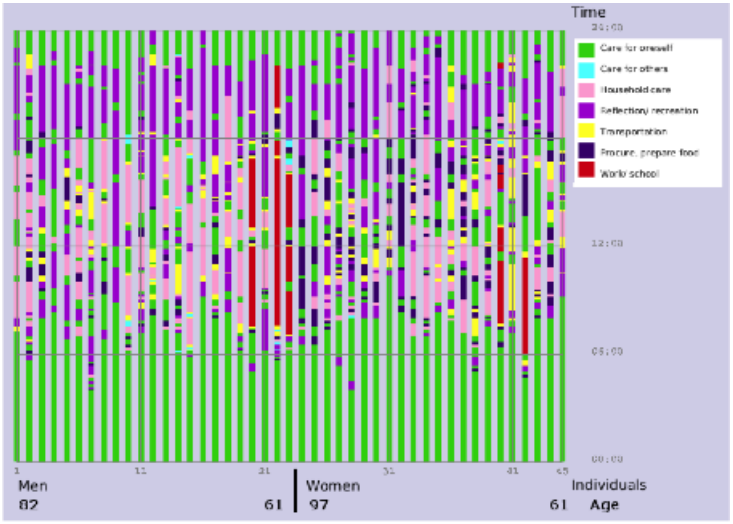


Figura 2.5: Atividades desenvolvidas ao longo dos dias

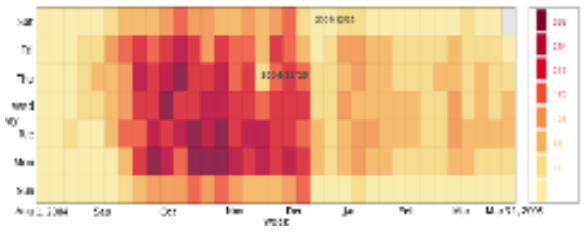


Figura 2.6: Padrões semanais de atividade laboral

Figura 2.7: Imagens de circuito de video-vigilância em diversas escalas temporais.

Figura 2.8: Pintando com corrida e *tracking* GPS

ao segundo e aumentando o intervalo de tempo na direção do centro em que o círculo mais central representa horas. Assim, de forma integrada, sem modificação de contexto, é possível avaliar o que acontece nas várias escalas de tempo com os dados existentes (ver figura 2.7).

Numa perspetiva mais abstrata e artística, o projeto YesYesNo [4], utilizou os dados recolhidos pela aplicação Nike+ de forma a criar pinturas com os registos GPS das corridas efetuadas por utilizadores da aplicação. Os resultados são visualmente apelativos e apesar da abstração ser elevada conseguem transparecer uma sensação de ritmo e velocidade de forma bastante clara e formando uma relação pessoal entre a visualização e o autor (reconhecimento da representação). Um dos exemplos gerados pode ser observado na figura 2.8.

2.3.2 Visualização de Grafos

A utilização de técnicas de visualização de grafos para representação de relações entre utilizações de locais importantes, pode ser um veículo importante para fazer emergir os padrões temporais a que um individuo está sujeito. Tal como referido em [CQ11], os grafos são técnicas de visualização poderosas pelas seguintes razões:

- Qualquer conjunto de dados com relações internas pode ser representado como um

[4] http://yesyesno.com/nike-collab-paint

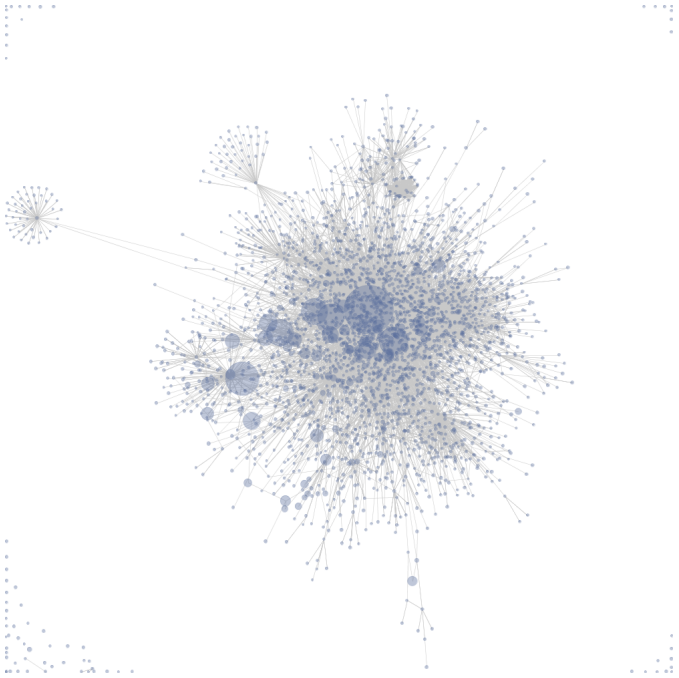


Figura 2.9: Disposição Spring-Layout da estrutura de links de uma wiki. Tamanho dos nós utilizado para representar a atividade de edição num dado dia.

grafo.

- Os utilizadores são dotados de fortes capacidades de processamento visual. A visualização de grafos pode envolver vários tipos de informação sobreposta e a soma visual dos seus elementos e relações permite a formulação de raciocínios visuais e a extração de nova informação.

- Uma das formas mais poderosas de representação de uma relação entre elementos consiste na união destes através de um segmento de reta.

A visualização de grafos levanta também algumas dificuldades inerentes à sua natureza. Grafos baseados em conjuntos de dados de grande dimensão podem provocar confusão visual por uma sobreposição excessiva dos seus elementos. Existe uma complexidade computacional envolvida na disposição de elementos em grafos que é incomportável para conjuntos de dados muito vastos, o que dificulta a sua utilização em contextos de necessidade de processamento em tempo real.

Na generalidade, para um pequeno número de nós um grafo é uma forma eficiente de transmitir relações presentes nos dados. A escolha do algoritmo de disposição das ligações e nós é também de importância extrema para a sua eficácia. Algumas regras genéricas são aceites como bases para uma boa visualização de um grafo. Entre estas destaca-se a tentativa de não cruzamento de ligações e a comparabilidade entre disposições para um grafo. Duas disposições para o mesmo grafo utilizando o mesmo mecanismo de arranjo devem ser o mais idênticas possível.

No contexto da presente dissertação é interessante analisar um tipo de algoritmo de disposição de grafos que pode ser interessante na tarefa de fazer emergir visualmente os padrões temporais da rotina de um indivíduo.

Spring Layout A técnica de Spring Layout, também denominada Force-Directed Layout consiste na modelação dos elementos e ligações que constituem um grafo como um sistema de partículas sujeitas a forças físicas. Consiste num algoritmo de minimização de uma função de energia (que pode variar de implementação para implementação) que iteração a iteração atribui a cada um dos nós uma nova posição. As ligações ente nós traduzem-se em forças de atração e repulsão que influenciam o resultado da função energia. Um exemplo deste tipo de disposição é apresentado na figura 2.9.

A utilização de grafos e técnicas de visualização associadas pode ser uma forma eficiente de permitir ao utilizador extrair informação dos dados sem que esta seja formalmente especificada.

2.4 Considerações Finais

No presente capítulo do documento foram apresentadas diversas técnicas e metodologias que formam uma base que permita a construção de uma solução para o problema proposto. A análise aqui efetuada é dividida pelos blocos considerados fundamentais para a construção da aplicação pretendida.

Inicia na secção 2.1 com a Captura e Processamento dos Dados que permitem a formulação de uma base de conhecimento, discutindo sistemas de localização e algoritmos de síntese e integração entre os domínios da localização e do tempo. De destacar o algoritmo de extração de Staypoints 2.3, bastante referido e estudado em diversas publicações e que permite, a partir de um conjunto de localizações físicas absolutas, extrair um conjunto de locais onde um utilizador permanece durante um período significativo, com vantagens e desvantagens bem documentadas. De assinalar ainda o sistema Geohash 2.1.3.3, que poderá ser bastante útil no processo de extração de informação, principalmente como forma de otimização dos algoritmos desenvolvidos através da criação de índices nos sistemas de armazenamento disponíveis em ambiente móvel.

Na secção 2.2 o tema abordado é a Extração de Informação à partir da base de conhecimento. Numa primeira fase referem-se as duas aproximações fundamentais para extração de locais com significado para o utilizador em 2.4 e 2.5 e apontam-se vantagens e desvantagens de ambas. Este é um tema fundamental para o desenvolvimento da modelação da rotina de um utilizador e apenas uma implementação fiável permitirá a obtenção de bons resultados através da utilização das técnicas apresentadas em 2.2.2. Esta temática, de grande abrangência e complexidade, carece de uma definição precisa dos pressupostos em que assenta e na definição clara do âmbito a que se quer sujeitar o conceito de rotina de um indivíduo. Esta tarefa é de tanta importância como a execução das técnicas em si, que geralmente se baseiam em modelos estatísticos.

O estudo de trabalho relacionado termina com a apresentação de técnicas de visualização consideradas relevantes e adequadas à tarefa de visualização de padrões temporais e ilustração de rotinas e diferenças. De destacar a efetividade de visualizações que utilizam a cor como forma de demonstrar a emergência de padrões como em 2.6 e também que se preocupam com a integração de horizontes temporais diversos na mesma visualização 2.7. Termina com uma visão geral sobre visualização de grafos, que constituem uma estrutura interessante no estabelecimento das relações entre os locais com significado para o utilizador, uma outra forma de fazer emergir a rotina deste através da forma como utiliza os locais com importância na sua vida.

Finalmente, de referir que no levantamento de trabalho relacionado aqui apresentado existiu a preocupação de apenas apresentar técnicas adequadas, ou passíveis de serem adaptadas, aos constrangimentos impostos pelo ambiente de computação móvel em que a aplicação será desenvolvida.

3

Visualização de Padrões Temporais : Ferramentas e Aplicação Móvel

O presente capítulo apresenta o conjunto de aplicações desenvolvidas no âmbito da dissertação, bem como uma discussão sobre opções de desenho, técnicas e conceptuais. Começa por apresentar os requisitos subjacentes ao projeto e um conjunto de aplicações acessórias. Termina com a apresentação da aplicação móvel, objetivo último do projeto.

3.1 Conceito e Requisitos

A presente secção tem como objetivo explicitar o conceito e os requisitos inerentes ao projeto Time Machine de forma a enquadrar adequadamente as secções que se seguem. Os temas abordados na dissertação resultam naturalmente da tentativa de materializar os conceitos e dar resposta ao requisitos que aqui se apresentam.

3.1.1 Conceito

O projeto Time Machine tem por objetivo explorar as potencialidades da computação ubíqua com o objetivo de revelar os padrões da vida quotidiana de um indivíduo. Pretende revelar estes padrões através de formas de visualização subjetivas e que criem uma relação empática com o utilizador. Uma cartografia pessoal que estimule a reflexão do utilizador acerca da sua forma de viver represente a rotina e assinale o especial, e ofereça um vislumbre do futuro. O desenvolvimento destas visualizações não é conceito único do projeto. A reflexão, durante o desenvolvimento, por parte do conjunto de intervenientes de diversas áreas como a arte, informática, e design, acerca das tecnologias

e dinâmicas subjacentes à computação ubíqua e à integração de diversas sensibilidades num projeto tecnológico, é também parte do núcleo do projeto. Time Machine não só pretende estudar o produto mas também o processo.

Partindo da recolha de informação sobre o posicionamento do utilizador no espaço, pretende explorar o telefone como objeto computacional mas também de afeto por parte dos utilizadores. Pretende desta forma responder às idiossincrasias do comportamento humano, por vezes rotineiro, por vezes imprevisível, utilizando algoritmos de inteligência artificial e testando os limites da computação móvel e ubíqua. O teste destes limites, de forma colaborativa com o utilizador, permite também que estes se tornem mais informados, ajudando-os na construção de um modelo mental adequado acerca da computação ubíqua e também de métodos de aprendizagem automática.

A comunicação através de técnicas de visualização subjetivas mas ao mesmo tempo legíveis e informativas, pretende servir de interface entre o utilizador e a aplicação. Cruzando sensibilidades artísticas e científicas, posiciona o projeto na área da arte multimédia.

3.1.2 Análise de Requisitos

Um conjunto de documentos foram realizados ao longo do desenvolvimento do projeto, contendo informação sobre a materialização dos conceitos anteriormente apresentados. Estes documentos foram criados de forma colaborativa por parte dos elementos envolvidos. No entanto, o facto da sua elaboração se ter efetuado durante as diferentes fases da execução, conduziu a uma alta volatilidade dos requisitos e a constantes alterações ao projeto e seu plano de desenvolvimento, resultando em alguma indefinição e num desenvolvimento de subprojetos que por vezes perderam contacto com a visão geral do projeto.

Através da análise destes documentos um conjunto de requisitos funcionais e de desenho foi identificado, os quais se expõem de seguida.

3.1.2.1 Requisitos de Desenho

Os requisitos de desenho surgem do conceito anteriormente descrito de oferecer uma comunicação reativa e subjetiva que mantenha uma ligação com o utilizador e que promova a reflexão do mesmo acerca do seu modo de vida. Os elementos visuais devem servir como peças de arte multimédia, criadas pela forma de viver do indivíduo. Seguidamente listam-se os requisitos de desenho principais a serem cumpridos pela aplicação.

- As visualizações devem ilustrar a diferença da rotina.
- As visualizações devem estimular a reflexão do utilizador e criar uma ligação com este (empatia visualização/dono da rotina).
- As visualizações devem suportar várias escalas de tempo, nomeadamente mensais, semanais e diárias.

- As visualizações devem permitir navegar sobre todo o histórico dos dados.
- As visualizações devem ser determinísticas.
- As visualizações devem ser atualizadas em tempo real, permitindo que o utilizador obtenha *feedback* visual imediato.
- As visualizações devem possuir uma componente de subjetividade e ser percecionadas como elementos de arte multimédia.

3.1.2.2 Requisitos Funcionais

Os requisitos funcionais identificados emergem naturalmente do conceito de modelação da rotina de um utilizador. Sendo este o requisito nuclear à volta do qual todos os outros gravitam, um conjunto de sub-funcionalidades são necessárias para o seu cumprimento. Uma área funcional diz respeito à identificação dos locais com significado para o utilizador, que servirão de matéria prima ao desenvolvimento da modelação da rotina. A análise dos elementos temporais e espaciais, nos diversos ciclos pretendidos (dia, semana, mês), encapsula outra área funcional. A predição/estimação do comportamento do utilizador encerra outro conjunto de funcionalidades pretendidas para a aplicação. Abaixo enumeram-se os principais requisitos funcionais identificados.

- A aplicação deve modelar a rotina de um utilizador através da captura de informação sobre o movimento do mesmo.
- A aplicação deve identificar locais percecionados como importantes para o utilizador.
- A aplicação deve detetar o local percecionado como casa pelo utilizador.
- A aplicação deve identificar situações em que o utilizador visita um novo local com significado.
- A aplicação deve conseguir classificar o dia como calmo ou agitado de acordo com a rotina do utilizador.
- A aplicação deve identificar uma utilização diferente de um local conhecido do utilizador.
- A aplicação deve identificar um dia extraordinário de acordo com a rotina do utilizador.
- A aplicação deve detetar padrões e suas quebras, tanto ao nível da utilização do tempo como dos locais.
- A aplicação deve procurar generalidade evitando assunções que excluam grupos de utilizadores.

- A aplicação deve ser dotada de um processo de aprendizagem maioritariamente não supervisionada, dispensando sempre que possível a colaboração do utilizador.
- A aplicação deve ser sensível a eventos ao nível da hora e da mudança de local, providenciando adaptação ao contexto.
- A aplicação deve prever o local para onde o utilizador se deslocará.
- A aplicação deve prever a utilização do tempo e locais no futuro próximo.
- A aplicação deve evitar ligações via rede celular.
- A aplicação deve apresentar um nível de disponibilidade elevado independentemente do contexto em que opera.
- A aplicação não deve prejudicar a utilização regular do telefone.
- A aplicação deve zelar pela privacidade do utilizador protegendo o acesso aos dados capturados e não os comunicando a qualquer entidade externa.
- A aplicação deve oferecer métodos de exportação dos dados capturados pelo utilizador.
- A aplicação de oferecer a possibilidade do utilizador reiniciar o processo de aprendizagem.
- A aplicação deve suportar a exportação das visualização para imagens digitais.

3.1.2.3 Plataforma de Desenvolvimento

Foi definido que a plataforma de desenvolvimento a utilizar seria a Android[1]. Várias razões levaram a esta escolha. Do ponto de vista do desenvolvimento é uma plataforma de desenvolvimento moderna que suporta todas as abstrações aceites como essenciais nos dispositivos móveis de última geração e não tem custos de desenvolvimento associados visto que o conjunto de ferramentas necessárias à criação de aplicações é gratuito. Do ponto de vista do projeto permite chegar a mais utilizadores, dado que é uma plataforma em franca expansão e a existência de dispositivos móveis de todos os segmentos de mercado que a suportam.

3.1.3 Considerações Finais

O levantamento de requisitos no contexto em que o projeto Time Machine se desenvolve é uma tarefa difícil. O foco do projeto assenta em conceitos subjetivos, como o conceito de rotina ou de local com significado para o utilizador, bastante distantes de definições próximas de implementação num sistema informático. Para além disso o projeto tem raízes experimentais e o facto de integrar diferentes sensibilidades dificulta por vezes

[1] http://www.android.com/

uma sintonia na compreensão dos conceitos e nas metodologias de trabalho adotadas. O cruzamento entre o mundo artístico e o da engenharia nem sempre flui da forma mais natural. Os métodos de trabalho, aliados à dificuldade de formalização de alguns conceitos, obrigam ao desenvolvimento de linhas de trabalho paralelas, muitas vezes tendo apenas como resultado final a especificação concreta de um conceito ou clarificação de um requisito.

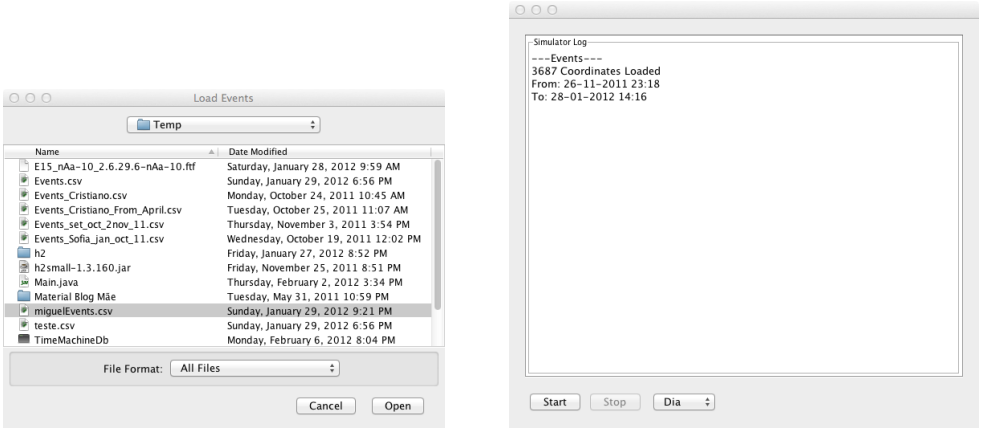
3.2 Simulador

No âmbito da dissertação de mestrado [dB11] uma aplicação Java foi criada com o objetivo de extrair os pontos de estadia e locais com significado para o utilizador à partir de um conjunto de ficheiros de texto contendo o registo de coordenadas recolhidas pela aplicação móvel. Esta aplicação segue uma aproximação *batch*, tirando partido do conhecimento prévio de todo o conjunto de dados para proceder à extração de informação sobre pontos de estadia e locais com significado para o utilizador através dos algoritmos apresentados em 2.3 e 2.4 no capítulo 2. Este processo é alvo de estudo detalhado na dissertação já referida tendo sido testado com dados reais, formando uma base sólida para a sua implementação na aplicação móvel. Este processo é fundamental para dar resposta a grande parte dos requisitos funcionais descritos e por isso considerou-se fulcral que a sua adaptação a uma aproximação em tempo real fosse garantidamente equivalente aquela que já se encontra estudada e da qual se conhecem vantagens e desvantagens. Este facto, aliado à necessidade de ter um protótipo com possibilidade de exportar informação para serem efetuadas algumas experiências visuais por parte de outros elementos do projeto, levou à criação de uma aplicação de simulação que permite a utilização de dados reais de forma iterativa, reproduzindo de forma ordenada o processo de captura.

3.2.1 Funcionalidade

O simulador desenvolvido toma como entrada um ficheiro em formato CSV (resultante da aplicação criada por Samuel del Bello em [dB11]) contendo a listagem ordenada temporalmente (mais antigo primeiro) de todas as coordenadas recolhidas. Este ficheiro é escolhido pelo utilizador depois do arranque da aplicação através de um menu de escolha de ficheiros. Depois de carregado, é apresentada uma janela com a informação sobre o horizonte temporal a que a simulação diz respeito dado o conjunto de dados de entrada, bem como o número de coordenadas carregado. As janelas iniciais podem ser vistas nas figuras 3.1(a) e 3.1(b) respetivamente para a escolha de ficheiro e informação inicial.

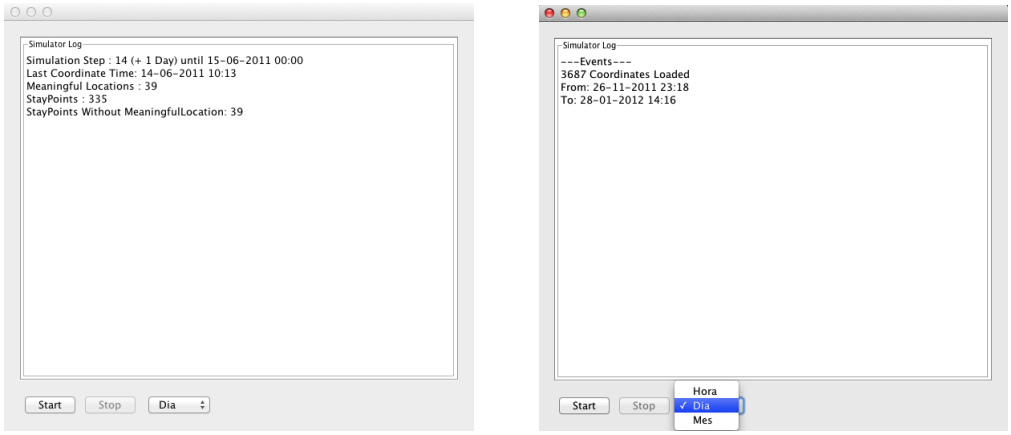
O utilizador pode então dar inicio à simulação clicando no botão Start. A cada passo de simulação, que pode ser respeitante a uma hora, um dia ou um mês, o simulador fornece iterativamente todas as coordenadas respeitantes a esse período de tempo ao sistema. O sistema por sua vez executa o algoritmo de extração de pontos de estadia e de locais importantes, bem como a modelação da rotina do utilizador a cada coordenada



(a) Escolha de ficheiro com coordenadas. (b) Janela com informação inicial.

Figura 3.1: Janelas iniciais Simulador

fornecida, fazendo persistir os cálculos numa base de dados relacional. A qualquer momento o utilizador pode parar a simulação e observar na janela alguns dados sobre a mesma, nomeadamente número de pontos de estadia, locais com significado e número de pontos de estadia não pertencentes a um local com significado. Enquanto a simulação está parada, é ainda possível efetuar uma cópia do ficheiro respeitante à base de dados relacional, contido na diretoria de execução, e inspecionar todos os dados colecionados até ao momento através da ferramenta de exploração fornecida para o sistema de gestão de base de dados. Neste caso, apesar de ser necessário o conhecimento da linguagem SQL, os dados podem ser manipulados com toda a expressividade oferecida. As janelas correspondentes a simulação parada pelo utilizador e escolha do período de simulação são apresentados nas figuras 3.2(a) e 3.2(b) respetivamente.



(a) Simulação parada pelo utilizador. (b) Escolha de período de simulação.

Figura 3.2: Janelas simulação

Quando terminada a simulação, o utilizador é notificado através da janela da aplicação (ver figura 3.3) e para além do ficheiro correspondente à base de dados relacional

referido anteriormente, um conjunto de ficheiros em formato textual é escrito, funcionado como ponte entre o simulador e qualquer aplicação que se pretenda desenvolver. Abaixo é apresentada uma listagem destes ficheiros, acompanhada de uma pequena explicação sobre cada um.

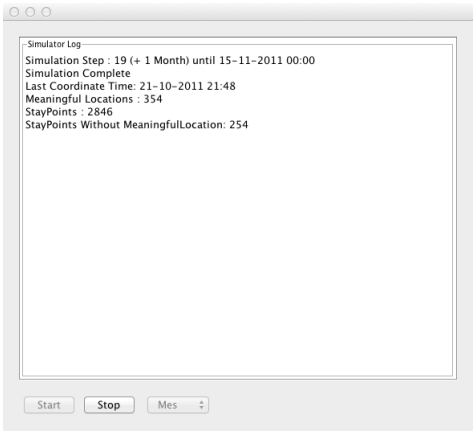


Figura 3.3: Janela simulação completa

locations.kml Ficheiro KML contendo todos os locais com significado, extraídos durante a simulação bem como os pontos de estadia associados a cada um. Pode ser aberto através da aplicação Google Earth possibilitando a visualização dos locais com significado para o utilizador sobre um mapa geográfico.

forcluster.arff Ficheiro ARFF usado pelo software Weka com informação sobre os locais com significado para o utilizador, nomeadamente frequência média de acesso, duração média de estadia, minuto médio de chegada, minuto médio de partida. Com este ficheiro foram realizadas diversas experiências de *clustering* de locais na tentativa de formar grupos que partilhassem as mesmas características.

container_id.csv Ficheiro em formato CSV contendo informação estatística sobre os dias no contexto de um modelo de utilização do tempo. Os atributos exportados são distância, número de mudanças de local, número de locais diferentes, duração de estadia em locais com significado, minuto da última chegada, minuto da primeira saída e número de novos locais. A finalidade destes ficheiros foi permitir a execução de experiências visuais relacionadas com a regularidade dos dias e a aferição da qualidade de modelação da rotina do utilizador durante o desenvolvimento.

context_id_morada.csv Ficheiro em formato CSV contendo informação estatística sobre os locais com significado para o utilizador no contexto de um modelo de uso do tempo. O ficheiro contém informação estatística diária sobre os atributos frequência, duração, minuto da última chegada no dia, minuto da primeira saída do dia e tempo entre visitas ao local. Estes ficheiros servem o propósito de permitir a inclusão da rotina de utilização dos locais tanto nas experiência visuais relacionadas com o dia como nas relacionadas com os acessos a locais.

network_data.csv Ficheiro em formato CSV contendo a informação absoluta e estatística dos acessos a locais de forma ordenada durante um dia. A informação absoluta exportada inclui minuto de chegada e saída do local, frequência total acumulada e duração total acumulada. A informação estatística registada é a mesma descrita para o ficheiro context_id_morada.csv . Estes ficheiros possibilitaram experiências visuais de visualização de grafos.

Numa versão inicial do simulador existiu ainda uma funcionalidade respeitante à tentativa de cálculo do dia efetivo do utilizador. A heurística utilizada para esta finalidade consistia na deteção de um período de estadia no local identificado como casa do utilizador superior a 5 horas indicando o fim do dia efetivo. O início do dia efetivo corresponderia à primeira saída do local identificado como casa do utilizador. O facto desta heurística não ser genérica, devido ao limite fixo de número de horas e de falhar em situações de dados em falta acabou por levar ao abandono da experiência. Em alguns dos casos analisados a heurística revelou-se funcional, permitindo transpor o ciclo solar de 24 horas e identificar o ciclo específico a que o utilizador estava sujeito.

3.2.2 Tecnologias Utilizadas

Para o desenvolvimento desta ferramenta foi escolhida a plataforma J2SE[2]. A escolha recaiu nesta plataforma devido à possibilidade de utilização da linguagem Java, a mesma linguagem utilizada na plataforma Android, o que oferece vantagem na portabilidade do código desenvolvido. Além disso, com apenas uma linha de desenvolvimento, é possível atingir independência do sistema operativo, permitindo que todos os elementos participantes no projeto o possam utilizar.

Duas bibliotecas externas foram utilizadas para o desenvolvimento do simulador. A biblioteca Joda-Time[3] para facilitar operações sobre datas. Estas operações são muito comuns na lógica da aplicação e a utilização de uma biblioteca externa, bem testada e com um leque alargado de abstrações promoveu a construção de código mais robusto. A biblioteca H2[4] descrita em 2.1.3.2, suportou a persistência dos dados no simulador. A opção de utilização desta biblioteca está relacionada essencialmente com dois fatores. O primeiro com a necessidade de utilização do modelo relacional dada a estrutura complexa dos dados. O segundo com a possibilidade de utilização da mesma em modo *standalone*, sendo a base de dados suportada por um único ficheiro em disco local, tal como se pretende na aplicação móvel final.

O conjunto de ferramentas fornecidas com a biblioteca H2 permite acrescentar funcionalidade ao simulador. A biblioteca inclui uma aplicação *web* que permite a conexão a bases de dados, estando ao dispor toda a expressividade SQL para manipulação das mesmas. Isto permite a qualquer momento da simulação inspecionar os dados presentes,

[2] http://www.oracle.com/technetwork/java/javase/overview/index.html
[3] http://joda-time.sourceforge.net/
[4] http://h2database.com/html/main.html

utilizando um navegador web. A figura 3.4 apresenta a aplicação web ligada a uma base de dados criada pelo simulador, com uma listagem de todos os locais com significado extraídos num determinado ponto da simulação.

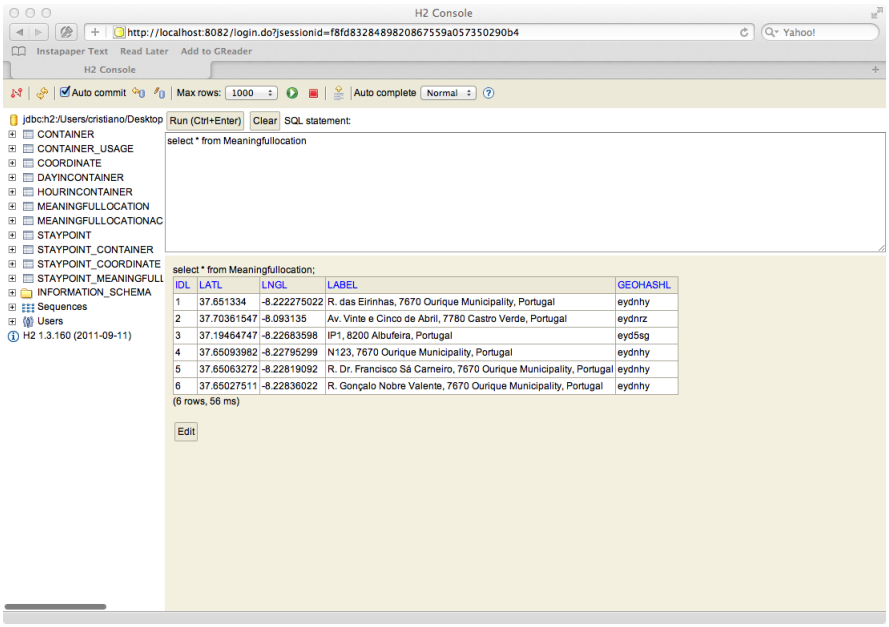


Figura 3.4: *Browser* do SGBD H2

De referir ainda que a biblioteca suportada oficialmente pela plataforma Android, SQLite, não possui nenhuma implementação fiável na plataforma J2SE, o que impossibilitou a sua utilização. Este facto não é grave visto ser possível abstrair programaticamente a camada de dados ou mesmo utilizar a própria biblioteca H2 na plataforma Android (ver 2.1.3.2). Já a portabilidade do ficheiro que contém a base de dados para a aplicação móvel, apenas é garantida no caso de utilização da mesma biblioteca, uma vez que o formato varia entre o SGBD SQLite e H2.

3.2.3 Considerações Finais

O simulador revelou-se uma ferramenta importante no decurso do desenvolvimento do projeto e da dissertação. Permitiu a realização de diversas experiências de forma mais ágil, sem perder contacto com o objetivo final de implementação da aplicação móvel. Apesar da evolução notável das ferramentas de desenvolvimento para plataformas móveis (Android incluído), o desenvolvimento com ferramentas *Desktop* continua a ser um pouco mais ágil, devido à maturidade das ferramentas de depuração e rapidez de desenvolvimento. A portabilidade do código é elevada visto que a linguagem de programação é partilhada entre a plataforma móvel e a de desenvolvimento do simulador tendo sido utilizadas poucas bibliotecas externas.

Com um pequeno esforço adicional é possível transformar esta ferramenta mais orientada a utilizadores experientes, numa ferramenta orientada ao utilizador final, permitindo a geração de um ficheiro com toda a informação extraída a partir de um conjunto de dados de localização referentes a um horizonte temporal alargado. Posteriormente este ficheiro poderia ser importado na aplicação móvel. Um caso de uso para esta funcionalidade seria a mudança de telefone por parte de um utilizador que assim poderia reaproveitar o seu anterior conjunto de dados, sem que a aplicação móvel tivesse de passar novamente pelo período de treino correspondente a uma utilização inicial.

Como ponto negativo no desenvolvimento desta aplicação reconheço alguma indecisão na escolha do SGBD utilizado no simulador. À altura do desenvolvimento não era claro, apesar de possível, se seria viável a utilização do SGBD H2 na plataforma Android. Apesar deste facto a inexistência de uma implementação J2SE da API SQLite da plataforma Android obrigava ao dispêndio de tempo na sua implementação, o que seria por si só um projeto de alguma dimensão, pelo que acabou por ser utilizado o sistema H2 tentando abstrair programaticamente a implementação concreta utilizada.

3.3 Visualizações Exploratórias

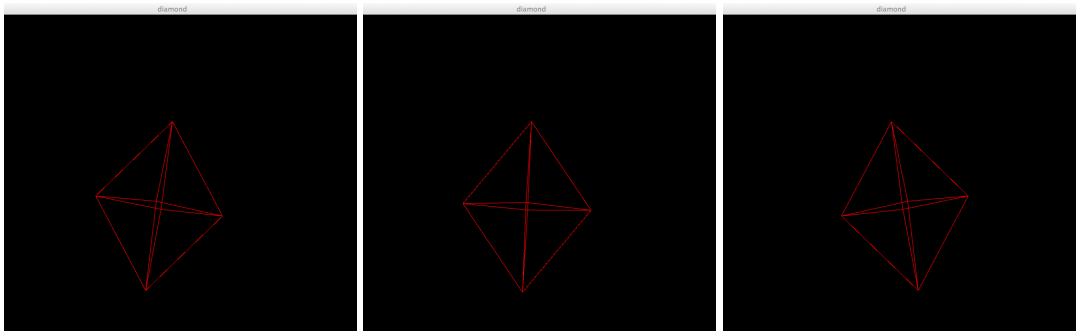
No decurso do projeto, e dada a multidisciplinaridade dos vários intervenientes, foi necessário avançar para o desenvolvimento de algumas visualizações exploratórias por forma a suportar um *feedback* visual concreto indispensável aos métodos de trabalho dos colaboradores da componente artística. Serve a presente secção para descrição das visualizações exploratórias desenvolvidas.

3.3.1 Visualização da Rotina e Atividade Através de um Objeto

Com o objetivo de expressar as noções de atividade e rotina ao longo do dia, foi desenvolvido um protótipo de visualização utilizando a plataforma de desenvolvimento Processing[5]. A ideia desta visualização é manipular um objeto de acordo com o nível de rotina e atividade a que um utilizador está sujeito. Tomando como ponto de partida duas pirâmides triangulares unidas pela base, como representação simplificada de um diamante, o objeto é sujeito a inclinações laterais e frontais ligadas ao nível de atividade e rotina, respetivamente.

Na figura 3.5 é possível ver de que forma o objeto é manipulado se o dia for completamente rotineiro (sem inclinação frontal) e com o nível de atividade a variar, de um dia mais frenético à esquerda para um dia mais calmo à direita. A atribuição da inclinação à esquerda para um dia menos ativo e à direita para um dia mais ativo, é de certa forma arbitrária. A relação que se tentou refletir é a de travagem para um dia menos ativo e de aceleração para um dia mais ativo. Na figura 3.6 é possível ver de que forma o objeto é afetado pelo nível de rotina, mantendo um nível de atividade fixo. O objeto tende a

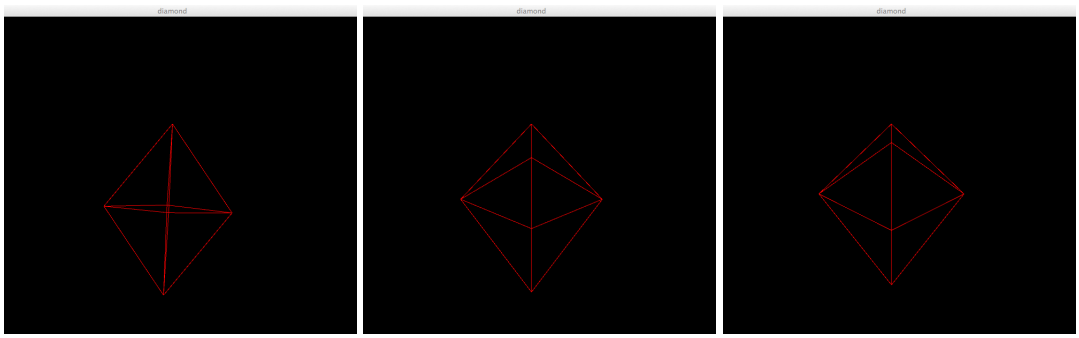
[5] http://processing.org/



(a) Dia muito ativo. (b) Dia normal. (c) Dia pouco ativo.

Figura 3.5: Dia rotineiro, nível de atividade variável.

inclinar-se frontalmente tanto mais quanto menos rotineiro for o dia. No limite, um dia totalmente fora da rotina reduzirá o objeto a um losango.



(a) Dia rotineiro. (b) Dia menos rotineiro. (c) Dia fora da rotina.

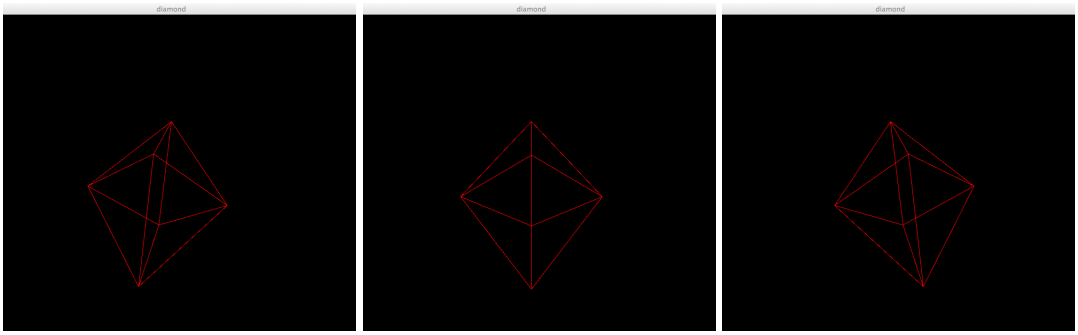
Figura 3.6: Nível de atividade regular (média), rotina variável.

O cruzamento das duas dimensões é apresentado na figura 3.7. A ideia de manipulação de um objeto através de eventos da vida do utilizador permite criar uma relação pessoal, uma certa empatia, dado que as suas ações se refletem diretamente no objeto, tornando-o seu.

A aplicação desenvolvida para teste da ideia de visualização aqui apresentada não chegou à ligação do objeto a dados concretos, tendo a manipulação sido efetuada através de eventos de teclado. Para além da manipulação do objeto, é ainda possível controlar o número de faces das pirâmides, a altura total e a cobertura das faces (*wireframe* ou cor sólida). Numa situação de ligação concreta aos dados, é necessário garantir que os conceitos de atividade e rotina não se intercetem (sejam independentes), por forma a salientar o cruzamento visual obtido.

3.3.2 Visualização da Novidade e Ritmo de Atividade do Dia

A visualização da novidade do dia e sua relação com a noção de ritmo para o utilizador, deram origem a um protótipo de visualização desenvolvido em Processing. Tirando



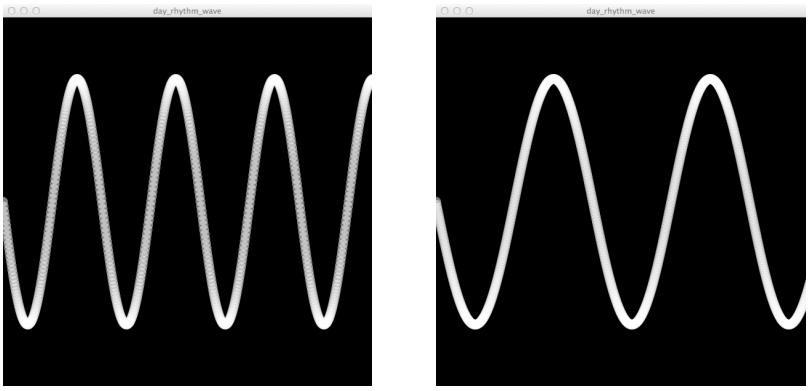
(a) Dia fora da rotina e muito ativo.

(b) Dia fora da rotina.

(c) Dia fora da rotina e pouco ativo.

Figura 3.7: Nível de atividade variável, fora da rotina.

partido de dados reais exportados pelo simulador, apresenta uma onda sinusoidal construída através do mapeamento entre a amplitude da onda e a percentagem de locais novos visitados pelo utilizador e entre o período e o tamanho do dia efetivo[6].



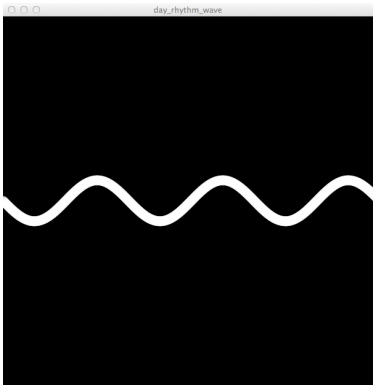
(a) Dia Efetivo = 218 m, Número de locais novos = 3, Número de locais conhecidos = 0

(b) Dia Efetivo = 555 m, Número de locais novos = 6, Número de locais conhecidos = 0

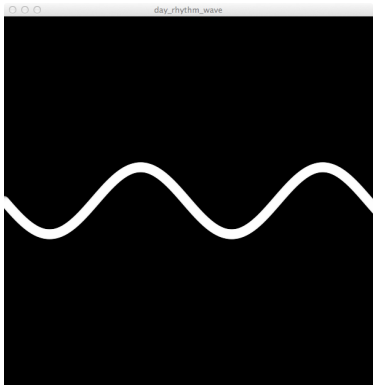
Figura 3.8: Onda de dias com amplitude de novidade igual e ritmos de atividade distintos.

O objetivo final da exploração deste tipo de construções visuais, prende-se com o desenvolvimento de fundos dinâmicos para futuras visualizações, que ofereçam ao utilizador um pulsar visual do ritmo e tipo de ritmo (novidade) a que está sujeito. A animação de uma onda deste tipo ao longo do dia, permite ao utilizador perceber que a aplicação móvel consegue percecionar a mesma ideia de ritmo que o utilizador tem do seu dia.

[6]Período decorrido entre a primeira saída e última chegada a um local com significado para o utilizador num dia.



(a) Dia Efetivo = 373 m, Número de locais novos = 1, Número de locais conhecidos = 5



(b) Dia Efetivo = 706 m, Número de locais novos = 3, Número de locais conhecidos = 8

Figura 3.9: Onda de dias com amplitudes de novidade e ritmos de atividade distintos.

3.3.3 Visualização do Dia Através de uma Rede de Acessos a Locais

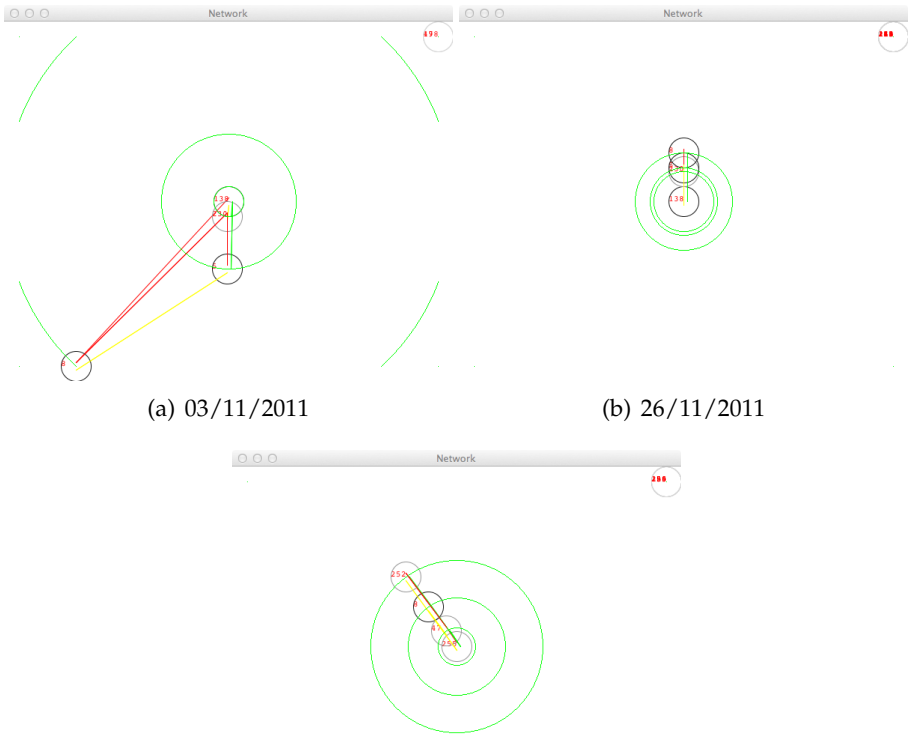
Da ideia de visualizar o dia, e o seu nível de rotina e novidade através da forma como o utilizador se move no espaço, surgiu o conjunto de aplicações aqui descrito. As aplicações desenvolvidas partilham o objetivo comum de visualizar os locais através de uma rede.

Dados reais suportam as visualizações, tirando partido da leitura de ficheiros CSV gerados pelo simulador descritos anteriormente (ver 3.2.1). Depois de carregados os dados, é feita uma simulação acesso a acesso, de forma a que a rede passe por todos os estados, tal como aconteceria no decorrer do dia na aplicação móvel.

3.3.3.1 Visualização Baseada em Distâncias Visuais

A ideia desta visualização passa por dispor os nós de forma a que os acessos a locais que são mais normais (mais rotineiros), ocupem uma posição central na janela. O nó mais próximo da rotina ocupa a posição central. Para todos os outros nós, é procurado o seu vizinho mais próximo já presente na visualização, em termos de semelhança da distância à rotina. Encontrado este vizinho, existe uma circunferência de pontos onde o nó pode ser colocado de forma a ficar à distância real correspondente, sendo escolhido aquele que cumpra dois requisitos : aproxime a distância real a todos os restantes nós presentes e esteja dentro da janela.

A dispersão visual dos nós, indica o quão diferente do normal foi o dia. Uma concentração de nós no centro da rede indica um dia em que os locais foram acedidos de acordo com a rotina do utilizador, ao passo que uma dispersão ao longo da janela indica um dia fora do normal. Este comportamento pode ser observado na figura 3.10. Claramente o dia representado em 3.12(b) foi o mais rotineiro, logo seguido do dia 3.12(c), sendo o menos rotineiro o dia 3.12(a).



(a) 03/11/2011 (b) 26/11/2011

(c) 03/12/2011

Figura 3.10: Visualização baseada em distâncias visuais: estado final da rede no fim do dia.

Na prática, depois de dispostos os dois primeiros nós, a especificação é cumprida dispondo todos os nós num segmento de reta (limitações da dimensão da janela à parte), visto que o posicionamento é feito segundo apenas uma dimensão dos dados (distância à rotina). Para os nós correspondentes a acessos a locais novos, é considerado que não possuem uma rotina, pelo que lhe é atribuída distância máxima aparecendo sistematicamente no canto da janela.

Apesar da distância entre nós ser o elemento nuclear da visualização, outros elementos visuais são utilizados para veicular informação. A listagem seguinte permite ler a visualização de forma completa.

Nó Representa o acesso a um local. Possui formato circular sem preenchimento, com a cor do traço a representar o grau de novidade do local. Um traço preto indica um local muito conhecido, cinzento escuro um local menos conhecido, e cinzento claro um local novo.

Ligação As ligações, apresentadas como traços de cor entre dois nós, representam a semelhança entre distâncias à rotina segundo um único atributo. Ligação vermelha implica distâncias à rotina semelhantes em termos de frequência de acesso, amarelo

em termos de duração de acesso e verde em termos de regularidade de acesso (hora de chegada e partida do local).

Etiqueta Identificador único do local. Sem significado para o utilizador, apresentado apenas para distinguir os locais para efeitos de teste.

Circunferência verde Utilizada visualmente para representar todos os pontos em que o nó poderia ser colocado de forma a ficar à distância correta do seu vizinho mais próximo.

3.3.3.2 Visualização Baseada em Algoritmos de Grafos

O protótipo seguinte foi desenvolvido no sentido de utilizar a estrutura grafo e suas técnicas de visualização como representação do dia do utilizador.

A aplicação desenvolvida possui três funcionalidades principais: construção do grafo a partir dos dados, visualização direta da rede nos seus diversos estados e exportação de dados referentes ao grafo que a suporta.

Construção do grafo Lendo um a um os acessos a locais presentes no ficheiro CSV de entrada, um grafo é criado para cada conjunto de acessos lidos até ao acesso corrente. Cada acesso é representado como um nó do grafo. Os nós são ligados a todos os outros que estejam no mesmo intervalo de distância à rotina. Três intervalos são utilizados, podendo os seus limites ser definidos num ficheiro de opções lido aquando do arranque da aplicação. Também a ligação entre nós correspondentes ao mesmo local pode ser ativada ou desativada nas opções. O utilizador pode ainda definir qual a frequência mínima acumulada que um local deve ter para que se considere que está sujeito ao conceito de rotina. Locais não sujeitos ao conceito de rotina não possuem, por omissão, ligações.

Exportação de dados A funcionalidade de exportação dos dados tem como objetivo a sua leitura por outras aplicações que disponibilizem diferentes visualizações. Para cada estado da rede, correspondente a um novo acesso a um local, é criado um ficheiro de texto de acordo com o formato utilizado pela aplicação *web* halfviz[7], contendo informação sobre nós, ligações e características visuais de ambos. Um exemplo de visualização numa ferramenta externa é apresentado na figura 3.11, em que o grafo respeitante a três dias distintos é apresentado segundo uma disposição baseada num algoritmo de forças físicas (ver descrição em 2.3.2).

Visualização A visualização direta dos vários estados da rede respeitantes a um dia é também funcionalidade da aplicação. O utilizador pode escolher entre três algoritmos de

[7] http://arborjs.org/halfviz/

(a) 03/11/2011

(b) 26/11/2011

(c) 03/12/2011

Figura 3.11: Visualização halviz: estado final da rede no fim do dia.

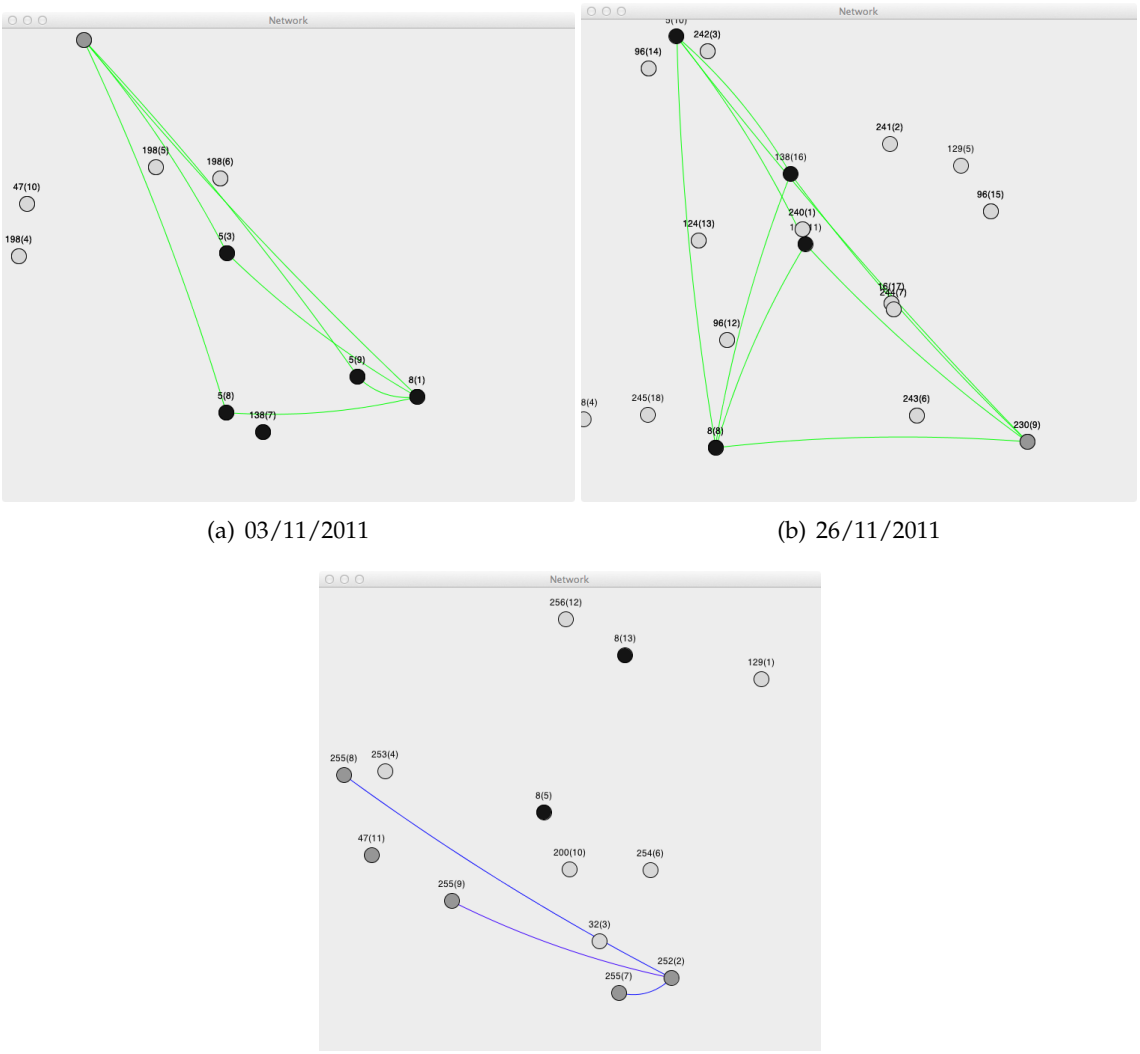
visualização de grafos suportados pela biblioteca JUNG[8]: Kamada-Kawai[9], Fruchterman-Reingold[10] e Spring Layout[11]. Um exemplo para três dias distintos, utilizando a visualização Fruchterman-Reingold, com a ligação entre acessos pertencentes ao mesmo local inativa, pode ser vista na figura 3.12. Para uma correta interpretação da visualização, é indispensável ainda a descrição dos vários elementos visuais da mesma. A listagem seguinte serve esse propósito.

[8] http://jung.sourceforge.net/

[9] http://jung.sourceforge.net/doc/api/edu/uci/ics/jung/algorithms/layout/KKLayout.html

[10] http://jung.sourceforge.net/doc/api/edu/uci/ics/jung/algorithms/layout/FRLayout.html

[11] http://jung.sourceforge.net/doc/api/edu/uci/ics/jung/algorithms/layout/SpringLayout.html



(a) 03/11/2011

(b) 26/11/2011

(c) 03/12/2011

Figura 3.12: Visualização através do algoritmo Fruchterman-Reingold: estado final da rede no fim do dia.

Nó Representa o acesso a um local. Possui formato circular com preenchimento. Preenchimento preto indica um local muito conhecido, cinzento escuro um local mediamente conhecido, e cinzento claro um local novo.

Ligação As ligações, apresentadas como traços de cor entre dois nós, indicam que estes se encontram no mesmo intervalo de distância à rotina. Uma ligação de cor verde indica uma pequena distância à rotina, azul uma distância média e vermelha uma distância assinalável. O utilizador pode definir os limites dos intervalos correspondentes a cada uma das cores, modificando dessa forma o grafo construído e a consequente representação.

Etiqueta Identificador único do local. Sem significado para o utilizador, apresentado apenas para distinguir os locais para efeitos de teste.

3.3.4 Considerações Finais

O conjunto de aplicações desenvolvidas permitiu uma experimentação visual concreta de ideias que foram surgindo ao longo do projeto. Permitiu aos elementos do projeto um contacto visual direto com algumas das suas ideias e em alguns casos uma certa integração dos dados que começavam a ser capturados e processados. Nesta altura a aplicação móvel não possuía maturidade suficiente para a prossecução destas experiências, pelo que o desenvolvimento de aplicações *Desktop* independentes foi a melhor solução encontrada.

A aplicação descrita em 3.3.1 permitiu uma experiência com os conceitos de rotina e atividade, através de uma visualização subjetiva, em que os efeitos produzidos pelo ato de viver do utilizador se refletem na manipulação de um objeto.

O protótipo apresentado em 3.3.2, tentou desenvolver o conceito de modelação visual do ritmo de novidade do dia através de uma onda sinusoidal. Este conceito pode ser interessante na criação de fundos de ecrã dinâmico, que reflitam em tempo real o ritmo a que o utilizador está sujeito ao longo do dia.

A utilização de grafos como forma de expressar relações entre elementos é de aplicação conhecida. Assim em 3.3.3.1 tentou modelar-se visualmente a rotina diária, como a expressão das relações entre os acessos a locais e a sua própria distância à rotina. Numa primeira aplicação os acessos a locais foram organizados de forma métrica, centralizando o nó com menor distância à rotina e dispondo os restantes em seu redor. A tentativa de respeitar as distâncias de forma precisa, levou a que os nós fossem dispostos em linha reta, o que não é visualmente atrativo, mas permitiu o desenvolvimento do conceito de órbitas de rotina inspirado precisamente nas circunferências verdes descritas anteriormente. Definindo o centro da janela como um lugar imaginário totalmente dentro da rotina, é possível dispor os locais acedidos durante o dia sobre as circunferências definidas pela sua distância à referência central. A definição destas órbitas pode ser feita para cada local ou para gamas de distâncias à rotina, caindo assim diversos locais na mesma órbita. A indicação dada pelas circunferências verdes da aplicação desenvolvida, revela um caminho visualmente interessante no sentido de expressar o tipo de dia pela largura das órbitas e o comportamento na utilização de cada local.

A utilização de grafos como estrutura de suporte, permitiu também a experimentação com algoritmos de visualização especificamente desenvolvidos para esse efeito. Para tal foi desenvolvida a aplicação descrita em 3.3.3.2, que permite a visualização do grafo de acesso a locais através de algoritmos de disposição dos nós, bem como a exportação da estrutura de suporte para visualização em aplicações externas. Apesar de parecer um caminho óbvio os resultados práticos não são totalmente satisfatórios. Este facto não parece estar relacionado com os algoritmos de visualização, mas antes com a modelação do problema através de grafos. A dificuldade de definição das relações que levam à ligação entre nós, principalmente pela sensibilidade dos intervalos de distância à rotina definidos e exclusão de locais não sujeitos a rotina, leva a que existam poucas ligações

entre nós. Estando os algoritmos de visualização de grafos implementados de forma a fazer emergir as relações entre nós e não a um arranjo espacial com significado, grande parte dos dias visualizados resultam num conjunto de nós dispersos na janela sem grande capacidade de transmitir informação.

3.4 Aplicação Móvel Time Machine

O desenvolvimento da aplicação móvel Time Machine é o objetivo último do projeto. Conduziu a diversas experiências já relatadas, guiando a presente dissertação no estudo e aplicação de técnicas que permitissem a satisfação dos requisitos levantados. A aplicação foi desenvolvida na plataforma Android[12], tal como expressado nos requisitos. A versão base de desenvolvimento foi a 2.2 que permite abranger a maioria clara dos dispositivos móveis que utilizam a plataforma. (ver 3.13).

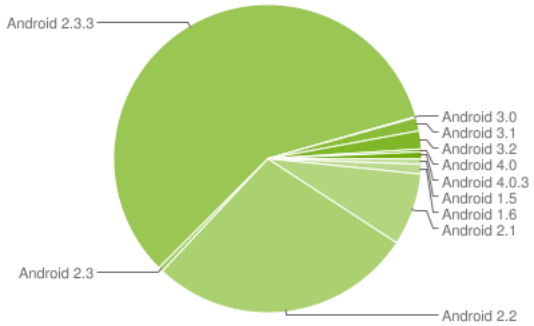


Figura 3.13: Distribuição de dispositivos por versão na plataforma Android (Março de 2012).

A presente secção do documento relata a aplicação móvel, descrevendo a funcionalidade principal segundo as principais áreas consideradas.

3.4.1 Captura e Processamento de Informação Geográfica

A aplicação desenvolvida captura de forma constante informação sobre a localização do dispositivo e consequentemente do seu utilizador. O processo de captura é iniciado automaticamente assim que a aplicação é lançada pela primeira vez ou já estando instalada, quando o telemóvel é ligado (no processo de *boot* do sistema). O utilizador não precisa de se preocupar em lançar a aplicação para que esta inicie o processo de captura da informação de localização.

A informação geográfica capturada é proveniente de dois serviços de localização distintos, conforme disponibilidade e qualidade da informação: via rede (celular e Wifi) ou GPS. A informação é comunicada pelo sistema operativo à aplicação, de acordo com os critérios indicados aquando do lançamento da mesma. O utilizador tem de permitir explicitamente nas definições gerais de sistema a captura de informação de localização a

[12] http://www.android.com/

partir de pelo menos uma das vias suportadas, caso contrário o sistema não a comunicará a qualquer aplicação instalada no telemóvel e consequentemente também a aplicação Time Machine não a receberá.

A informação obtida é armazenada utilizando dois suportes distintos. No armazenamento interno do telemóvel, através do registo das coordenadas capturadas numa base de dados SQLite. Utilizando o armazenamento externo, vulgo cartão de memória, em formato ficheiro de texto simples, com uma coordenada a ocupar uma linha e um ficheiro a corresponder a um dia. A funcionalidade de armazenamento externo necessita do suporte físico onde os ficheiros são escritos, o que nem sempre acontece. O suporte físico pode ser inexistente dependendo do telemóvel (a plataforma não obriga à existência de um), ou mesmo existindo estar indisponível pelo facto de estar ligado como disco a um computador. Nos restantes casos os ficheiros são escritos com a informação capturada, providenciando uma forma de armazenamento que garante máxima portabilidade e facilidade de acesso por parte do utilizador.

Seguidamente à captura e armazenamento da informação de localização, é efetuado um processamento da mesma de forma a extrair os pontos de estadia do utilizador (pequenas vizinhanças onde o utilizador permanece durante algum tempo), sendo o resultado guardado na base de dados existente no armazenamento interno.

A aplicação permite que o utilizador consulte de forma rápida toda a informação associada à ultima coordenada capturada, através de uma interface gráfica. A informação apresentada é a seguinte: latitude e longitude da coordenada capturada, precisão associada, data e hora de captura, sistema de localização utilizado, morada associada e uma lista dos satélites GPS visíveis no momento juntamente com a força do sinal de cada um. A figura 3.14 permite observar um estado concreto da interface para consulta de informação relativa à última coordenada capturada.

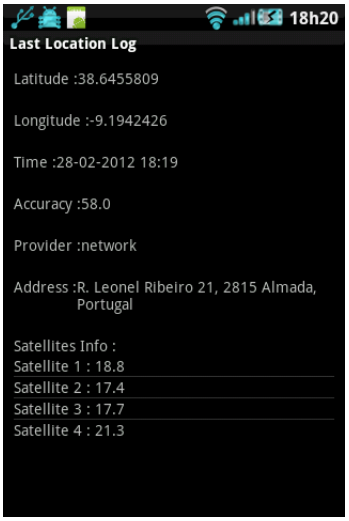


Figura 3.14: Interface gráfica para consulta de informação relativa à última coordenada capturada.

3.4.2 Extração de Informação

A extração de informação a partir dos dados capturados é executada pela aplicação em tempo real. Para cada ponto de estadia detetado é executado o processo de extração de informação relacionado com os locais com significado para o utilizador. O resultado deste processo pode ser a identificação de um novo local, fusão de locais existentes, ou simplesmente afetação do ponto de estadia ao local a que pertence. Esta informação é registada na base de dados interna e serve de base à modelação de rotina efetuada pela aplicação. Em paralelo, um conjunto de medidas estatísticas sobre o dia, e o acesso a locais com significado para o utilizador é registado na base de dados. A união da informação estatística e dos locais com significado para o utilizador formam o modelo de rotina pessoal, respondendo ao principal requisito da aplicação.

O utilizador pode consultar parte da informação extraída pela aplicação, nomeadamente sobre locais com significado, consultando a interface gráfica Clustering. Nesta interface pode exportar todos os locais detetados até ao momento, juntamente com os pontos de estadia associados, para um ficheiro KML no armazenamento externo. Esta funcionalidade é acessível através do botão Export Data e o ficheiro resultante pode ser aberto na aplicação Google Earth[13] para visualização sobre um mapa geográfico. A lista de locais pode ainda ser consultada em tempo real, sendo apresentada informação sobre o identificador interno do local e a palavra correspondente ao seu *geohash*. Este último pode ser utilizado no serviço `http://geohash.org/` para obtenção da localização aproximada do mesmo. Nesta interface o utilizador tem ainda a possibilidade de reiniciar a aprendizagem da aplicação, carregando opcionalmente os dados presentes num ficheiro contido no armazenamento externo, cujo conteúdo deve ser uma lista de coordenadas capturadas de acordo como o formato utilizado no Simulador. Esta funcionalidade é acedida clicando no botão Do Clustering.

Na figura 3.15 é possível ver a interface apresentada para oferecer a funcionalidade descrita anteriormente.

3.4.3 Visualização

Um conjunto de visualizações, algumas na linha das experiências descritas em 3.3, outras mais desenvolvidas e próximas de versões finais, foram integradas na aplicação móvel Time Machine. Estas visualizações tiram já partido do conhecimento da aplicação sobre o utilizador, partindo de dados concretos recolhidos, para apresentar representações com uma componente subjetiva e capaz de estimular reflexão de acordo com os requisitos da aplicação.

[13] `http://www.google.com/earth`

Figura 3.15: Interface para consulta de informação aos locais com significado extraídos.

3.4.3.1 Distância Diária Relativa

Nesta atividade o utilizador pode navegar ao longo de todos os dias registados pela aplicação, visualizando a distância percorrida por dia, relativamente ao máximo alguma vez registado. A navegação é feita através de manipulação táctil, podendo o utilizador arrastar a linha do tempo para a direita ou esquerda com o dedo, ou clicar numa das barras para a centrar no ecrã. A navegação pode ser feita de forma mais rápida ou precisa, suportando o conceito de aceleração no arrastamento da linha do tempo. Esta funcionalidade é bastante útil, visto que dado o tipo de aplicação se prevê que a linha do tempo cresça bastante permitindo ao utilizador saltar vários dias de uma vez.

A altura das barras e o gradiente de cor são utilizados como forma de expressar a distância percorrida. Uma barra mais alta e próxima do vermelho significa distância maior. A barra correspondente ao vermelho puro e altura máxima corresponde à maior distância alguma vez percorrida pelo utilizador num dia. A linha de tempo indica o dia concreto a que a barra diz respeito e a sua cor o tipo de dia: verde representa um dia de fim de semana, vermelho de semana.

Na figura 3.16 é possível observar um exemplo da atividade aqui descrita, no período de 01/06/2011 a 05/06/2011 com a distância máxima a ser observada na sexta feira dia 03/06/2011.

3.4.3.2 Visualização Forma Geométrica

Com o intuito de expressar o nível de atividade a que o utilizador está sujeito ao longo do dia foi desenvolvida uma visualização partindo da ideia explorada em 3.3.1. A ideia é transmitir ao utilizador o quão regular está a ser o seu dia em termos de atividade. A quantidade de atividade por tempo decorrido desde o início do dia é utilizada para calcular o tipo de dia que o utilizador esta a ter. Esta relação é literalmente transmitida

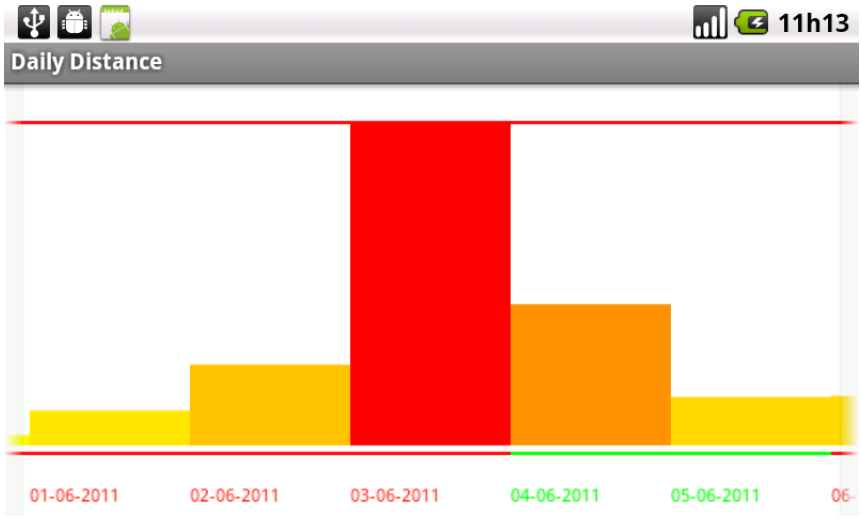


Figura 3.16: Interface para visualização da distância diária relativa percorrida pelo utilizador.

utilizando um quadrado e a deformação dos seus ângulos internos. A quantidade de atividade e a facilidade com que o quadrado se deforma está relacionada com a atividade habitual a que o utilizador está sujeito. Um utilizador que tenha uma vida mais sedentária terá mais facilidade em deformar o seu quadrado, sendo recompensado pelo seu esforço. Já para um utilizador habitualmente mais ativo o quadrado terá mais dificuldade em se modificar, estimulando o utilizador a ser ainda mais ativo. Num caso de atividade extrema o quadrado desaparecerá, indicando um nível de atividade superior, estando este comportamento correlacionado com um novo máximo de atividade por período de tempo para o utilizador.

Um exemplo (3.17(a)) de alguma atividade e outro (3.17(b)) de pouca atividade matinal, pode ser verificado nas figuras correspondentes.

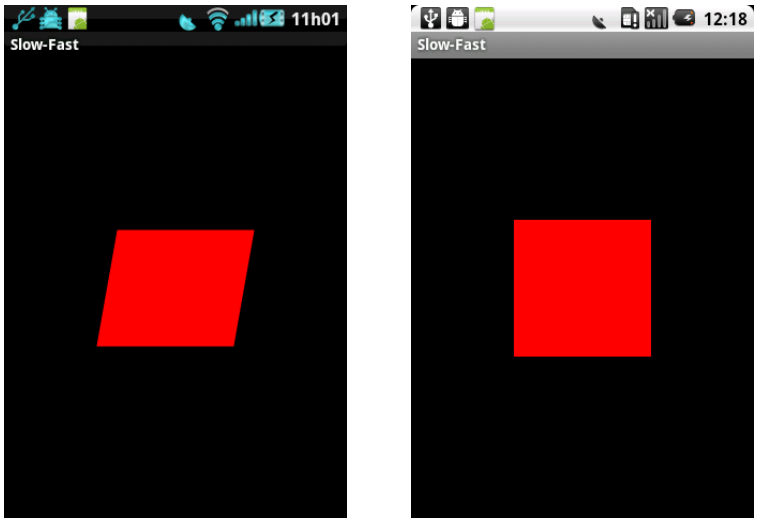
3.4.3.3 Qual é a cor deste Local?

Esta visualização foi desenvolvida como teste de reatividade da aplicação em tempo real às coordenadas capturadas. A atividade pinta o ecrã com uma cor, na gama entre amarelo e vermelho, para permitir ao utilizador ter *feedback* sobre a familiaridade do local em que se encontra. Uma cor mais próxima do vermelho indica um lugar mais desconhecido, por oposição a uma cor mais próxima do amarelo que indica um local mais familiar.

A visualização é pouco elaborada como pode ser observado através da figura 3.18, mas apresenta uma responsividade próxima do tempo real, modificando a cor a cada coordenada capturada e providenciando *feedback* o mais imediato possível.

3.4.3.4 Quadrados de familiaridade

A visualização quadrados de familiaridade, é o desenvolvimento do conceito apresentado em (3.4.3.3). Através da cor de dois quadrados inscritos num terceiro pretende-se



(a) Uma manhã não muito quadrada (bastante atividade).

(b) Uma manhã quadrada (nenhuma atividade).

Figura 3.17: Visualização Forma Geométrica.

em tempo real indicar ao utilizador a familiaridade do sítio onde se encontra. O quadrado exterior representa o quão familiar é o local corrente de acordo com o histórico total do utilizador na escala de familiaridade de todos os locais conhecidos. O quadrado intermédio indica a mesma informação considerando apenas o histórico referente ao fim de semana ou dias de semana. Já o quadrado interno representa esta informação relativamente ao histórico do dia de semana (segunda, terça, . . . , domingo) em que o utilizador se encontra. Os quadrados são coloridos numa escala de cinzento, com branco a representar o local mais familiar (mais brilho) e preto um local desconhecido (menos brilho).

De acordo com esta representação um local totalmente novo dará origem a um grande quadrado preto e um que seja considerado o mais conhecido em todos os históricos aplicáveis, um grande quadrado branco. O único elemento gráfico adicional utilizado é uma moldura azul, de forma a separar o fundo de ecrã da representação propriamente dita. Alguns exemplos desta visualização são apresentados na figura 3.19 com a respetiva explicação do seu significado.

3.4.3.5 Pequenos Retângulos de Tempo

Com o intuito de incluir nas visualizações desenvolvidas uma componente de previsão do comportamento do utilizador, foi desenvolvida a presente visualização, que permite apresentar o nível de atividade horária em relação à media semanal observada, e para além disso uma previsão do que acontecerá no restante do dia atual e nos dois seguintes. A visualização assemelha-se a um horário semanal, em que cada retângulo indica o nível de atividade comparativamente com a média da semana corrente. A cor do retângulo é utilizada como forma de expressar a normalidade da hora. Um retângulo mais próximo

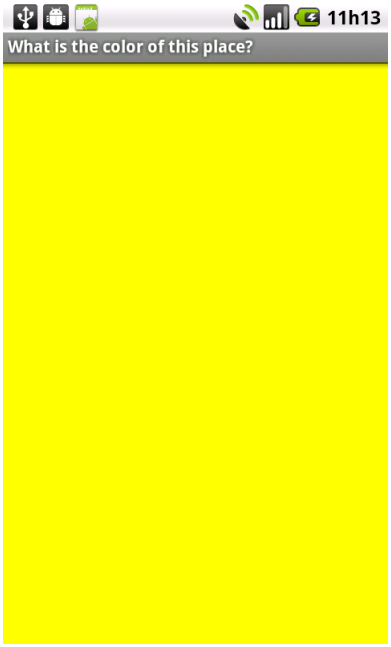


Figura 3.18: Visualização cor do local de acordo com a sua familiaridade.

do preto indica uma hora mais ativa que o habitual e o mais perto do branco um comportamento mais de acordo com a rotina semanal. A rotina de atividade é assim expressa segundo um período semanal, com o utilizador a ser confrontado com um vislumbre do futuro. Na figura 3.20, pode observar-se a semana passada de um utilizador, bem como o vislumbre do futuro previsto, com a separação entre os tempos efetuada pela barra de gradiente vermelho-amarelo. Dois exemplos para outro utilizador são apresentados na figura 3.21, para duas horas diferentes do mesmo dia.

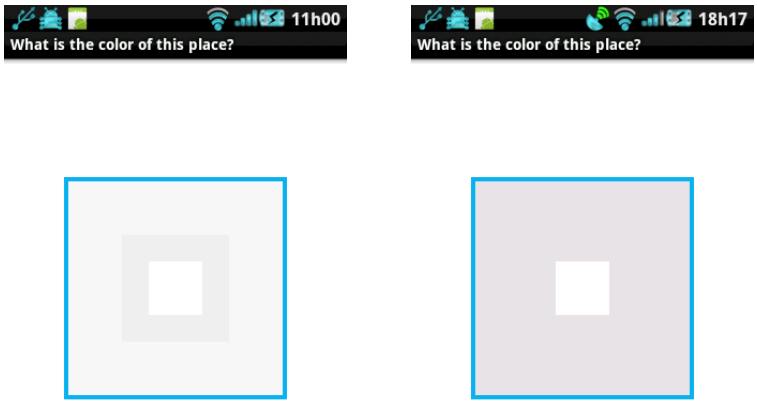
3.4.4 Interface

A interface desenvolvida é bastante simples, não tendo sido alvo de desenvolvimento aprofundado. Consiste numa simples lista de botões que permite o acesso às diversas visualizações e restantes funcionalidades. A lista dos botões é apresentada assim que a aplicação é lançada, ficando à disposição do utilizador para que este escolha a visualização pretendida. Esta interface pode ser visualizada na figura 3.22.

3.4.5 Tecnologias Utilizadas

Durante a descrição das diversas funcionalidades desenvolvidas várias tecnologias foram sendo referidas. A presente secção sistematiza a sua listagem apresentando considerações acerca da utilização das mais relevantes.

Como referido anteriormente, a plataforma de desenvolvimento móvel utilizada foi a Android, com o requisito mínimo de versão 2.2. Esta versão é considerada aquela que consolidou a plataforma como uma plataforma de desenvolvimento estável e com bom desempenho. As diferenças para a versão 2.1 são muito relevantes, dada a utilização de



(a) O local mais familiar nas segundas, não tão familiar nos dias de semana e dos mais familiares no histórico total.

(b) O local mais familiar às quartas, e relativamente não tão familiar durante a semana e no histórico total.

Figura 3.19: Visualização quadrados de familiaridade.

uma nova máquina virtual[14] que possibilita ganhos de desempenho notáveis. Adicionalmente, e como apresentado anteriormente na figura 3.13, esta versão permite incluir a maioria dos dispositivos móveis comercializados atualmente.

A linguagem oficialmente utilizada no desenvolvimento de aplicações para esta plataforma é a linguagem Java, sendo oferecido um conjunto de APIs que suportam as funcionalidades comuns no contexto do desenvolvimento de aplicações móveis. De relevar as APIs de localização, que permitem a obtenção dos dados utilizados pela aplicação Time Machine provenientes de dois fornecedores distintos: GPS e localização via rede (Wifi e Celular). As APIs de localização oferecidas abstraem completamente os sistemas de localização utilizados, oferecendo às aplicações apenas a informação de localização concreta pretendida sem esforço de desenvolvimento adicional.

Para o desenvolvimento das visualizações foi utilizada a API de desenho 2d fornecida pela plataforma. O desenho é efetuado através das primitivas gráficas oferecidas, de conhecimento comum, e transversais à maioria das plataformas que permitem desenho 2d. No desenvolvimento da interface foi utilizada a linguagem XML oferecida para descrição do posicionamento e arranjo dos elementos gráficos.

O armazenamento em formato relacional é efetuado através do SGBD SQlite, descrito em 2.1.3.1, o sistema oficialmente suportado pela plataforma, e para o qual é oferecida uma extensa API de suporte às operações de leitura e escrita necessárias.

Foi ainda utilizada uma biblioteca externa, Joda Time (3.2.2), para suporte de operações relacionadas com datas e tempos. Esta biblioteca é bastante conhecida e suportada na comunidade de desenvolvimento Java, sendo considerada robusta e eficiente. É a

[14] http://en.wikipedia.org/wiki/Dalvik_(software)

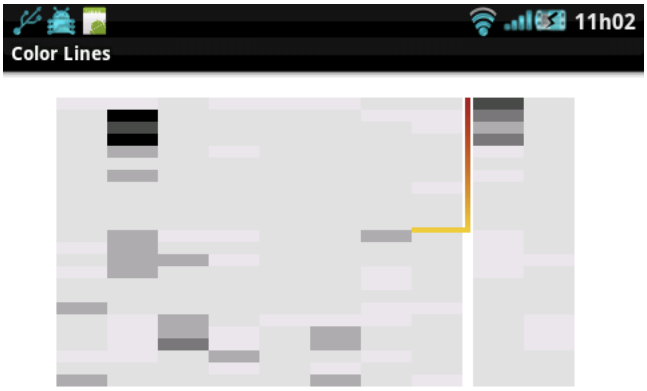
Figura 3.20: Semana de um utilizador, prevê-se um futuro de atividade bastante rotineira (cinzento).

única dependência externa da aplicação, no entanto justificável pelo fraco suporte da linguagem Java e da própria plataforma Android para lidar com abstrações temporais.

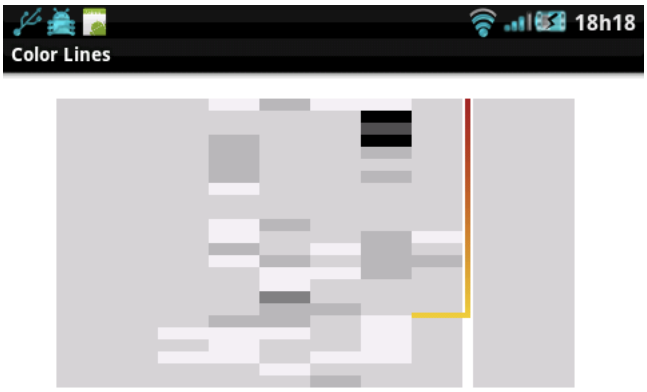
3.4.6 Considerações Finais

A aplicação Time Machine é o objetivo final do projeto. Foi possível desenvolver um conjunto sólido de funcionalidades, principalmente no que diz respeito à captura e extração de informação. A nível de visualizações foi implementado um conjunto que servisse de prova de conceito respondendo ao mesmo tempo aos requisitos visuais identificados. De destacar as visualizações 3.4.3.2, 3.4.3.4 e 3.4.3.5 que são bons exemplos do espírito do projeto e tiram já partido dos dados recolhidos pela aplicação em tempo real. Promovem reflexão questionando o utilizador (3.4.3.2), indicam comportamentos especiais permitindo ao utilizador o reconhecimento do extraordinário (3.4.3.4) e até um vislumbre do futuro (3.4.3.5). Tudo isto suportado por um sistema complexo, que integra a recolha e processamento de localização geográfica em paralelo com algoritmos de extração de informação sobre os locais com significado para o utilizador e sua rotina.

Como ponto negativo gostaria de realçar o fraco desenvolvimento da interface da aplicação, que consiste apenas num conjunto de botões que permitem o acesso às diversas visualizações.



(a) Semana de um utilizador: 11 horas do dia corrente.



(b) Samana de um utilizador: 18 horas do dia corrente.

Figura 3.21: Visualização pequenos retângulos de tempo.

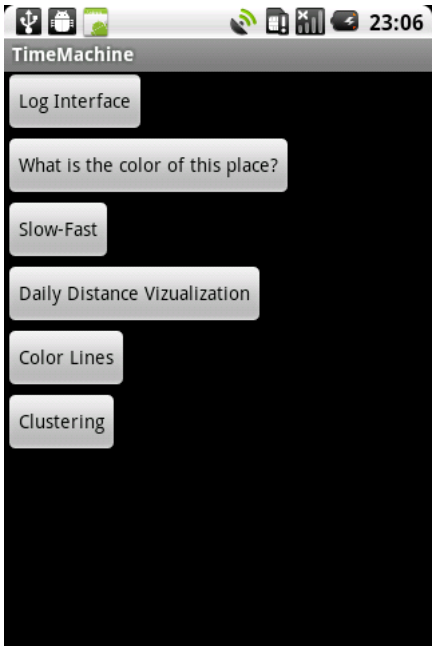


Figura 3.22: Interface inicial da aplicação.

4

Realização do Sistema

Neste capítulo apresentam-se os aspetos mais relevantes da solução desenvolvida como resposta aos requisitos aplicacionais levantados. São expostas as dificuldades encontradas na realização e a forma como foram ultrapassados. Pretende-se assim contribuir o conhecimento obtido que se espera seja importante na abordagem de problemas semelhantes, com especial foco na aplicação móvel mas aflorando também as ferramentas de suporte.

4.1 Aplicação Móvel Time Machine

O desenvolvimento da aplicação Time Machine pode ser dividido em duas fases distintas. A primeira surge da necessidade imediata de captura de dados para construção de um conjunto robusto de teste. Este conjunto de dados permitiu o estudo e desenvolvimento de algoritmos de extração de informação a partir de dados reais, não só no âmbito da presente dissertação mas também de outra anterior [dB11]. A segunda fase consiste na integração e adaptação dos algoritmos desenvolvidos externamente ao contexto móvel, no desenvolvimento da arquitetura da aplicação e de um conjunto de visualizações consentâneas com os requisitos visuais identificados.

A descrição efetuada de seguida apresenta uma visão geral da solução desenvolvida e a forma como os diferentes blocos funcionais são integrados e se relacionam. Seguidamente uma descrição mais detalhada da implementação de cada um dos blocos funcionais é efetuada completando a exposição da solução.

4.1.1 Android

Tal como referido em 3.1.2.3 a aplicação foi desenvolvida na plataforma Android. O desenvolvimento de uma aplicação na plataforma Android é feito através de componentes. Os componentes dividem-se em quatro categorias distintas: Content Providers, Activities, Services e Broadcast Receivers. Destes, apenas os últimos três foram efetivamente utilizados no desenvolvimento da aplicação tornando-se relevante a sua descrição.

Activity Representa uma interface com o utilizador. Uma aplicação é composta por diversas Activities que possuem independência relativamente a todas as outras mas devem formar uma experiência coesa para o utilizador. A independência entre Activities possibilita até que sejam invocadas por uma aplicação diferente daquela a que pertencem. Uma Activity é implementada estendendo a classe Activity e implementado os métodos da superclasse necessários para dar resposta aos requisitos da aplicação. A aplicação contém uma Activity principal, que é usualmente utilizada como ponto de entrada na primeira vez que é lançada. Uma Activity pode então lançar outras, sendo o sistema responsável por manter uma pilha de atividades que permite a navegação para a anterior. A Activity está sujeita a um ciclo de vida, em que cada evento corresponde à invocação de um método que o assinale. O ciclo de vida a que uma Activity está sujeita é apresentado na figura 4.1.

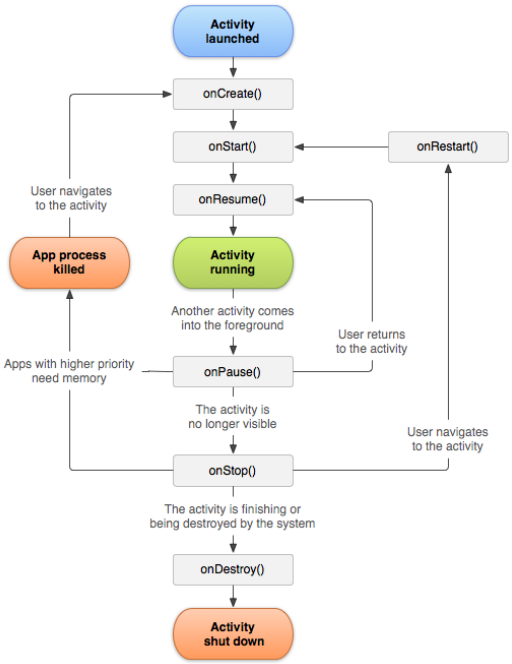


Figura 4.1: Ciclo de vida de uma Activity

Services Um Service corresponde a um componente responsável por efetuar trabalho em *background* sem providenciar uma forma de interação com o utilizador. Um componente da aplicação pode lançar um Service de modo a realizar trabalho em

background não sendo este interrompido mesmo que o utilizador mude para outra aplicação. Um componente de aplicação pode ligar-se a um Service e interagir invocando métodos expostos por este, segundo um modelo cliente-servidor.

Um Service pode assumir duas formas: *started* e *bound*. Em modo *started* o Service tem de ser lançado por um componente de aplicação através do método `startService`, podendo executar indefinidamente. O próprio Service é responsável por se auto-destruir quando tiver completado a operação que implementa. Em modo *bound* o Service é criado quando algum componente de aplicação se liga a este. Neste caso o serviço é destruído quando não existe mais nenhum componente ligado. Apesar da distinção entre estas duas formas de operação, um serviço *started* pode também permitir ligações de outros componentes depois de lançado.

Os Services não executam na sua própria *Thread* podendo causar bloqueio na interface de utilizador da aplicação a que pertencem. O próprio Service sempre que realize alguma operação demorada deve responsabilizar-se por lançar uma *Thread* para executar a operação, mantendo assim a capacidade de resposta da interface.

Também os Services estão sujeitos a um ciclo de vida, resultando na execução de métodos de *callback* a cada evento correspondente. Uma descrição deste ciclo de vida e métodos associados pode ser observado na figura 4.2. Para desenvolver um Service basta estender a classe Service e implementar os métodos da superclasse que o programador ache conveniente para a solução pretendida.

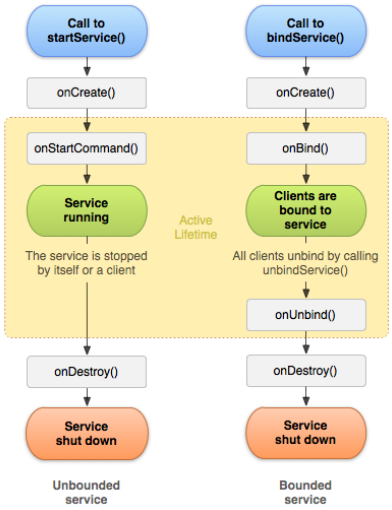


Figura 4.2: Ciclo de vida de um Service

Broadcast Receivers O componente Broadcast Receiver permite implementar resposta a eventos difundidos pelo sistema, anunciados a todas as aplicações. Este tipo de componente não possui uma interface direta com o utilizador embora possa beneficiar da barra de notificações para assinalar a difusão de um evento. Os eventos podem ser de sistema ou definidos pelas aplicações, devendo estas assinalar

que pretendem reagir a determinado tipo de evento. Para além deste registo devem implementar a lógica que pretendem executar sempre que o sistema anuncie um evento de um tipo para o qual se tenham registado através da criação de um subclasse da classe BroadcastReceiver. Implementando os métodos pretendidos da superclasse (usualmente o método `onReceive`) devem lançar um componente responsável pela execução de uma determinada operação.

Para que o sistema possa validar a aplicação e invocar os seus diferentes componentes é necessário desenvolver um ficheiro descritivo em formato XML denominado Manifest.xml contendo informação sobre a aplicação. A descrição de todos os componentes pertencentes à aplicação é obrigatória e contém informação como o nome do componente, permissão de invocação por outras aplicações e classe Java correspondente à implementação do componente para referir apenas alguns exemplos. No entanto, a informação contida neste ficheiro não se limita aos componentes pertencentes à aplicação. Um conjunto de informação referente à aplicação deve também ser incluído neste ficheiro, como o nome da aplicação, o seu ícone, versão mínima da plataforma, tamanhos e resoluções de ecrã suportados e um conjunto de permissões que a aplicação requer para funcionar. No momento da instalação o utilizador é notificado acerca das permissões que a aplicação pretende obter, devendo então decidir prosseguir ou não com a sua instalação. Estas permissões permitem aceder a serviços de localização, sensores do dispositivo, serviços de rede entre outros.

A plataforma Android define ainda uma linguagem específica, novamente expressa através de um ficheiro XML, para definição da interface gráfica da aplicação. Cada janela apresentada pela aplicação corresponde, como referido anteriormente, a um componente Activity. A plataforma Android permite associar um ficheiro XML contendo todos os elementos gráficos e seu arranjo a uma Activity separando assim interface e lógica da aplicação. Apesar desta funcionalidade a plataforma permite que uma Activity implemente dinamicamente o seu arranjo gráfico o que é bastante útil na criação de visualizações que não tiram partido dos elementos gráficos base (*widgets*).

4.1.2 Arquitetura Aplicacional

A arquitetura da aplicação é baseada nos componentes anteriormente definidos, e pode ser descrita como sendo uma arquitetura de três camadas. A camada de apresentação é implementada através de componentes do tipo Activity que por sua vez interagem com a camada de controlo para sua atualização. A camada de controlo é responsável pela implementação de toda a lógica da aplicação. Estabelece a relação entre os dados e a sua apresentação e entre o sistema de captação de localização e a aplicação. O processamento e extração de informação a partir dos dados de localização capturados é efetuada através de *Managers*, implementados segundo o padrão de desenho Singleton que se dividem em dois blocos principais: Clustering Manager e Time Container Manager. Existe ainda um

bloco responsável pela resposta aos eventos de localização gerados pelo sistema, implementado através de um componente do tipo Service e um bloco Analysis para tratamento de informação a ser servida à camada de apresentação. A camada de dados é composta por três blocos principais: a base de dados relacional utilizada para o armazenamento de toda a informação capturada e extraída pelos *Managers*, o sistema de ficheiros para registo de informação num formato independente da plataforma para ser lido por outras aplicações e o fornecedor de localização gerido pelo sistema operativo Android.

Na figura 4.3 é possível ver esquematizada a arquitetura da aplicação segundo os principais blocos existentes nas várias camadas.

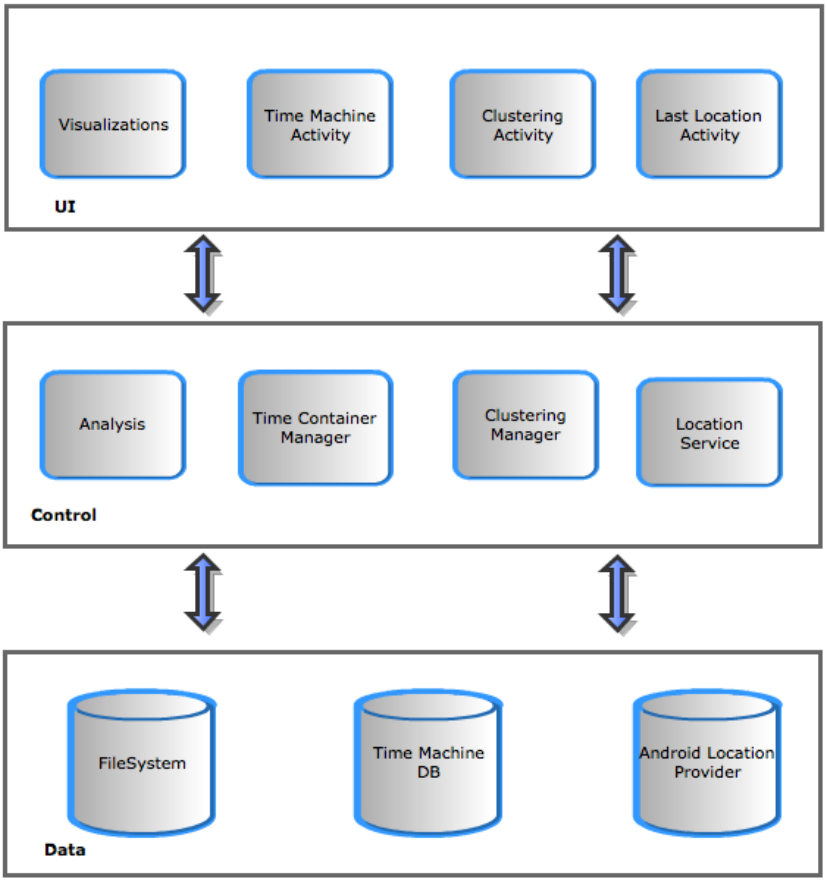


Figura 4.3: Arquitetura da aplicação Time Machine

A utilização deste tipo de arquitetura permitiu que o desenvolvimento dos *Managers*, o núcleo da aplicação, fosse independente do sistema Android, possibilitando o seu desenvolvimento no Simulador e assim acelerando o processo de desenvolvimento e de testes a partir dos dados já capturados. Além disso os benefícios deste tipo de arquitetura, nomeadamente a separação de *concerns*, permite uma aplicação mais robusta e fácil de manter dado o carácter modular do desenvolvimento.

O ponto de entrada na aplicação é a Activity Time Machine, a partir da qual o utilizador pode aceder às diversas Activities disponibilizadas. Como ponto de entrada, tem de ser declarada no ficheiro Manifest.xml de forma especial, possibilitando assim que o sistema saiba a raiz da aplicação e possa efetuar a gestão da pilha de Activities. A aplicação é responsável pelo lançamento dos diversos componentes individuais que a constituem mas está automaticamente habilitada a beneficiar do comportamento de navegação através do *back button* existente em todos os dispositivos Android, que tira partido precisamente desta pilha.

Dado o seu estatuto especial é nesta Activity que são executadas algumas configurações iniciais necessárias ao funcionamento da aplicação.

Na primeira execução após a instalação da aplicação é criado no armazenamento externo a estrutura de diretorias utilizada para as diversas funcionalidades da aplicação. A estrutura utilizada é bastante simples como se pode observar em 4.1. No diretoria Bd são guardados *backups* da base de dados relacional em formato SQLite, sempre que o utilizador utilize a funcionalidade de importação de dados a partir de um ficheiro CSV contendo uma listagem de coordenadas. O ficheiro a ser carregado deve estar presente na raiz da estrutura de diretorias da aplicação ou seja na diretoria TimeMachine com o nome Events.csv. A diretoria Kml é utilizada para colocar os ficheiros KML contendo a informação dos locais com significado para o utilizador e pontos de estadia, exportados a pedido do utilizador. As coordenadas capturadas pela aplicação são armazenadas na diretoria Coordinates, um ficheiro por dia com o nome a designar o dia a que os dados pertencem.

Listing 4.1: Estrutura de diretórios da aplicação no armazenamento externo

```
1 /TimeMachine
2 /TimeMachine/Bd
3 /TimeMachine/Kml
4 /TimeMachine/Coordinates
```

Outra das ações desenvolvidas na primeira execução é o lançamento do serviço responsável por atender eventos gerados pelos fornecedores de localização. Este serviço informa o sistema que pretende receber informação acerca da localização do dispositivo e regista-se como *listener* deste tipo de eventos.

A Activity na sua primeira execução é também responsável pela criação da base de dados relacional utilizada para suportar o armazenamento de dados evitando assim um comportamento de *lazy initialization*. A criação da base de dados é feita através da leitura de um ficheiro contendo o *script* SQL e da subsequente execução dos comandos nele contidos. O ficheiro contendo o *script* é incluído como recurso no pacote da aplicação possibilitando a sua leitura dinâmica e também o *deployment* de atualizações ao modelo de dados através da modicação do ficheiro.

A aplicação Time Machine implementa dois fluxos aplicacionais principais, um despoletado por um evento de nova coordenada, outro pelo utilizador. Quando uma nova

coordenada é entregue ao bloco Location Service pelo fornecedor de localização de sistema, este tem a responsabilidade de a entregar ao bloco Clustering Manager que efetuará a deteção de Staypoints (pontos de estadia). Caso um Staypoint tenha sido terminado, procede seguidamente à extração de informação respeitante aos locais com significado para o utilizador, podendo o novo Staypoint ser adicionado a um local existente, dar origem a um novo local ou à fusão de locais existentes. O Staypoint terminado é então entregue ao bloco Time Container Manager que procederá a dois tipos de análise : análise de acesso a locais com significado e análise diária. A primeira consiste essencialmente na deteção e registo de acessos a locais com significado e a segunda à recolha e registo de medidas associadas ao dia permitindo assim a construção de um pequeno repositório de dados a partir do qual será feita a análise à rotina do utilizador. A figura 4.4 ilustra o fluxo descrito.

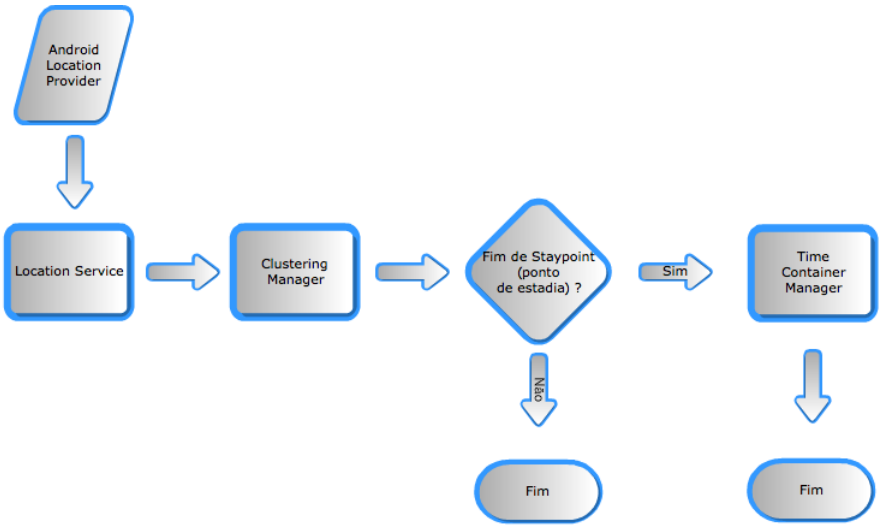


Figura 4.4: Fluxo nova coordenada detetada.

O fluxo iniciado por um evento com origem no utilizador ocorre com o lançamento de uma Activity associada a uma visualização. Depois do pedido de visualização ser efetuado, a Activity vai obter a informação que permita executar a expressão visual pretendida. Para isso cada Activity possui o seu próprio bloco Analysis que contacta para obter a informação necessária. O bloco Analysis responde conforme requerido analisando o contexto atual (usualmente a localização atual) procedendo à entrega os dados necessários à visualização. A Activity passa então ao desenho propriamente dito, oferecendo ao utilizador a resposta visual pretendida dado o contexto.

4.1.3 Modelo de Dados

O modelo de dados desenvolvido é composto por dois módulos. O módulo relacional consiste numa base de dados implementada através do sistema SQLite, utilizada para o armazenamento dos dados capturados e extraídos de forma estruturada permitindo uma manipulação complexa. O módulo baseado em ficheiros de texto serve o propósito do

armazenamento da unidade atómica da aplicação, a coordenada, a partir da qual todos os restantes dados podem ser reconstituídos. O formato textual garante maior portabilidade que o relacional podendo ser manipulado em qualquer sistema informático de forma simples.

A visão geral do modelo de dados relacional pode ser observada na figura 4.5. Do diagrama de entidades e relações foram omitidos os atributos das entidades no sentido de melhorar a legibilidade. De seguida, apresenta-se uma descrição detalhada de cada uma das entidades e dos respetivos atributos, contextualizando a sua utilidade no âmbito da aplicação.

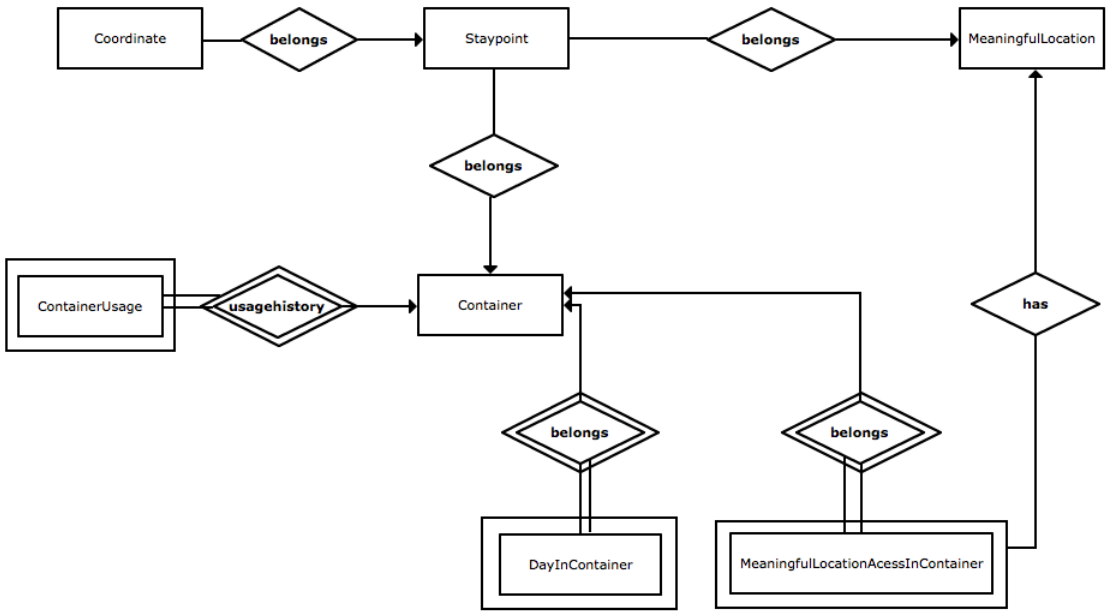


Figura 4.5: Diagrama de entidades e relações.

4.1.3.1 Coordinate

A entidade Coordinate é utilizada para efetuar o registo de todas as coordenadas capturadas. A coordenada é o elemento atómico de toda a aplicação e apesar de não ser diretamente utilizada para a modelação da rotina é a base do processo de extração de informação. É a entidade que se espera mais dados contenha com o decorrer do tempo, mas não é utilizada diretamente para quaisquer *queries* a não ser a obtenção esporádica da última coordenada registada quando a aplicação arranca pela primeira vez depois de um reinício do sistema. Esta entidade serve basicamente como *backup* dos ficheiros de texto que são também escritos com todas as coordenadas capturadas.

idc Identificador único que distingue a coordenada, gerado automaticamente. Chave primária da entidade.

latc Latitude da coordenada.

lngc Longitude da coordenada.

provider Fornecedor de localização do sistema que indicou a coordenada: Rede ou GPS.

accuracy Precisão associada à captura da coordenada indicada em metros. Um valor mais alto indica pior qualidade.

dtc Tempo associado à captura da coordenada. A informação horária diz respeito à data, hora e minuto em que a coordenada foi capturada.

geohashc *Geohash* com precisão de aproximadamente 2.4 km correspondendo a uma cadeia de carateres de comprimento 5 que indica a região retangular a que pertence a coordenada.

4.1.3.2 Staypoint

O Staypoint é a unidade mais importante no processo de extração de informação que a aplicação executa. É uma entidade que possui relação com a grande maioria das entidades do modelo. Aqui são armazenados todos os Staypoints detetados pela aplicação a partir das coordenadas recolhidas numa região espacial confinada durante um período de tempo. Um Staypoint pode pertencer a uma só MeaningfulLocation (local com significado para o utilizador), simbolizando tempo passado nesse local, e possivelmente uma chegada ou partida deste. Possui pelo menos uma coordenada ficando assim explicada a sua relação com a entidade Coordinate, e uma relação com a entidade Container estando associado obrigatoriamente a um modelo de uso de tempo. Esta relação possuí um atributo não chave, correspondendo à distância percorrida (em km) desde o último Staypoint. A quantidade de dados esperada nesta entidade é bastante elevada e ao contrário da entidade Coordinate são efetuadas *queries* de alguma complexidade, pelo que é necessário algum cuidado na sua otimização.

ids Identificador único que distingue o Staypoint, gerado automaticamente. Chave primária da entidade.

lats Latitude média do Staypoint.

lngs Longitude média do Staypoint.

dtstart Data, hora e minuto de início da estadia do utilizador no Staypoint.

dtend Data, hora e minuto do fim de estadia do utilizador no Staypoint.

geohashs *Geohash* com precisão de aproximadamente 2.4 km correspondendo a uma cadeia de carateres de comprimento 5 que indica a região retangular a que pertence o Staypoint.

4.1.3.3 MeaningfulLocation

Entidade que representa os locais com significado para o utilizador (MeaningfulLocation). Este locais são obtidos através de um processo de *clustering* de Staypoints pelo que cada MeaningfulLocation possui pelo menos um Staypoint associado. A entidade possui ainda relação com MeaningfullocationAcessInContainer indicando o local a que o acesso diz respeito. Esta tabela tem alguma volatilidade sendo sujeita a mais operações de atualização e remoção que as restantes. Este facto advém do processo de *clustering* descrito em 4.1.5.1. A quantidade de dados esperada é bastante inferior às anteriores entidades.

idl Identificador único que distingue a MeaningfulLocation, gerado automaticamente. Chave primária da entidade.

latl Latitude média dos Staypoints associados ao local.

lngl Longitude média dos Staypoints associados ao local.

label Morada associada à coordenada média da MeaningfulLocation.

geohashl *Geohash* com precisão de aproximadamente 2.4 km correspondendo a uma cadeia de carateres de comprimento 5 que indica a região retangular a que pertence a MeaningfulLocation.

nsps Número de Staypoints associados ao local. Este atributo é redundante visto que poderia ser obtido através da contagem de tuplos presentes na relação entre Staypoint e MeaningfulLocation. No entanto por razões de desempenho este valor é armazenado, permitindo o cálculo incremental da coordenada média da MeaningfulLocation de cada vez que um novo Staypoint é associado. Tirando partido da formula de cálculo da média o processo é efetuado em $O(1)$ em vez de $O(n)$ a custo da atualização e registo de um atributo extra.

4.1.3.4 Container

Container é a entidade que representa o modelo de utilização do tempo detalhado em 4.1.5.2. Os atributos desta entidade são apenas três mas a sua importância reside na relação com as entidades DayInContainer, MeaningfulLocationAcessInContainer, ContainerUsage e Staypoint. Destas relações emerge o conceito de rotina do utilizador. É esperada uma quantidade de dados diminuta, visto que não é expectável que mesmo o utilizador menos rotineiro ultrapasse as dezenas de modelos de utilização do tempo em que vive.

idcnt Identificador único que distingue o Container, gerado automaticamente. Chave primária da entidade.

active Indica se o Container está a ser utilizado.

geohashcnt *Geohash* com precisão variável representado por uma cadeia de carateres variável. Define a região considerada normal para movimentação do utilizador. Funciona como etiqueta geográfica do modelo de utilização do tempo e é utilizada para detetar uma transição para um modelo de utilização de tempo diferente. Detalhe em 4.1.5.2.

4.1.3.5 ContainerUsage

Entidade fraca que representa o histórico de utilização dos diversos modelos de utilização do tempo (Containers). A chave primária da entidade é a união entre idcnt e datestartusage. Para além de permitir um conhecimento sobre os modelos de tempo utilizados ao longo do tempo, indica também o fluxo entre Containers que é importante para que o sistema consiga perceber para que modelo de utilização do tempo deve transitar.

idcnt Identificador do Container a que a utilização diz respeito.

datestartusage Data, hora e minuto correspondentes ao inicio da utilização do Container.

dateendusage Data, hora e minuto correspondentes ao final da utilização do Container.

previouscnt Identificador do Container anterior a esta utilização.

trig Evento (*trigger*) que originou o fim da utilização do Container.

4.1.3.6 MeaningfulLocationAcessInContainer

A entidade fraca MeaningfulLocationAcessInContainer representa os dados associados ao acesso a um local com significado para o utilizador, de forma a permitir a extração de informação sobre estes no contexto de um modelo de utilização do tempo. A chave primária é composta pelos atributos idcnt e starttime.

idcnt Identificador do Container em que o acesso ocorreu.

starttime Data, hora e minuto de inicio de acesso ao local.

endtime Data, hora e minuto de final de acesso ao local.

weekday Identificador numérico (1-7) do dia da semana correspondente ao início do acesso ao local. Apesar de ser informação redundante permite uma filtragem eficiente dos dados segundo o dia da semana.

duration Duração total em minutos do acesso ao local com significado.

arrival Minuto do dia correspondente à chegada ao local com significado.

leave Minuto do dia correspondente à partida do local com significado.

idl Identificador único do local a que o acesso corresponde.

4.1.3.7 DayInContainer

Entidade fraca composta por um conjunto de atributos respeitantes a valores observados durante um dia do utilizador num modelo de utilização do tempo. A chave primária da entidade é composta pela união dos atributos idcnt e day. Esta entidade é utilizada para extrair valores estatísticos associados ao dia no contexto de um Container.

idcnt Identificador do Container em que o dia ocorreu.

day Data do dia.

weekday Identificador numérico (1-7) do dia da semana associado a day. Dados redundantes, atributo incluído pelas razões já referidas.

numberofmeaningfullocations Número de locais com significado diferentes visitados num dia.

numberoftransitions Número de mudanças de local durante um dia.

durationinmeaningfullocations Duração de permanência em locais com significado ao longo do dia em minutos.

firstarrival Primeira chegada a um local com significado no dia (minuto do dia em que ocorre).

lastarrival Última chegada a um local com significado no dia (minuto do dia em que ocorre).

firstleave Primeira saída de um local com significado no dia (minuto do dia em que ocorre).

lastleave Última saída de um local com significado no dia (minuto do dia em que ocorre).

distance Distância percorrida no dia expressa em quilómetros.

numberofnewplaces Número de novos locais com significado visitados durante o dia.

4.1.3.8 Índices

As entidades mais sujeitas a *queries* são aquelas que estão envolvidas nos dois blocos de extração de informação da aplicação. A implementação do processo de *clustering* é baseada em *queries* de proximidade geográfica que podem ser bastante otimizadas tirando partido deste facto. O sistema SQLite na plataforma Android não oferece qualquer capacidade de criação de índices multidimensionais, os únicos relevantes para otimização de *queries* de proximidade geográfica. Para contornar esta situação cada entidade envolvida no processo de *clustering* possui um atributo *geohash* (ver 2.1.3.3) que demarca a região geográfica a que o elemento pertence. Utilizando este atributo é possível filtrar os dados de forma a que a *query* analise apenas os tuplos geograficamente relacionados com a

query a executar. Para otimizar esta filtragem foi criado um índice para todos os atributos *geohash* tirando partido da unidimensionalidade dos mesmos na representação dos pares de coordenadas geográficos.

A extração de informação sobre a rotina do utilizador é baseada num modelo estatístico construído sobre as entidades DayInContainer e MeaningfulLocationAcessInContainer. Este modelo estatístico envolve *queries* que filtram os dados segundo diferentes atributos temporais para depois procederem à agregação estatística. No sentido de otimizar estas filtragens foram criados índices em ambas as tabelas sobre o atributo `weekday`. Na entidade MeaningfulLocationAcessInContainer foi também criado um índice sobre o atributo `idl` de forma a otimizar a filtragem e agrupamento por local com significado para o utilizador.

4.1.3.9 Modelação Objeto-Relacional

Todos os objetos que possuem correspondência a entidades do modelo de dados implementam a interface Java DBObject apresentada em 4.2.

Listing 4.2: Interface DBObject

```
1  public interface DBObject {
2    public void setId(long id);
3    public String getTableName();
4    public ContentValues getContentValuesInsert();
5    public ContentValues getContentValuesUpdate();
6    public String getWhereForUpdate();
7    public String[] getWhereUpdateValues();
8    public String[] getWhereRemoveValues();
9    public String getWhereForRemove();
10 }
```

Esta interface permite abstrair as operações CRUD (*create*, *replace*, *update*, *delete*) dos objetos que possuem uma relação direta com as entidades do modelo de dados. A utilização desta interface permite melhorar a modularidade do sistema seguindo uma filosofia de separação de *concerns*, ficando cada classe responsável pela implementação da sua forma de criação, atualização e remoção na base de dados. Além disso permite que a quantidade de código desenvolvido seja inferior e a sua legibilidade superior pois em qualquer parte do sistema são utilizados os mesmos métodos para criação, remoção e atualização de objetos na base de dados. Estes métodos são apresentados em 4.3.

Listing 4.3: Métodos CRUD

```
1    public void insertDBObject(DBObject dbo) {
2      long id = getWritableDatabase().
3            insert(dbo.getTableName(), null, dbo.getContentValuesInsert());
4      dbo.setId(id);
5    }
6    public void removeDBObject(DBObject dbo) {
7      getWritableDatabase().delete(dbo.getTableName(),
```

```
8                       dbo.getWhereForRemove(),
9                       dbo.getWhereRemoveValues());
10    }
11    public void updateDBObject(DBObject dbo) {
12      getWritableDatabase().update(dbo.getTableName(),
13                      dbo.getContentValuesUpdate(),
14                      dbo.getWhereForUpdate(),
15                      dbo.getWhereUpdateValues());
16    }
```

Na inserção depois de obter a base de dados para escrita, o método pede o nome da tabela onde deve inserir o novo tuplo e os pares chave-valor a inserir. O valor devolvido pela inserção corresponde ao identificador único do tuplo (`idc, ids, idl, idcnt`) que é então atualizado no objeto acabado de inserir mantendo consistência entre a base de dados e o objeto em memória. A atualização e remoção são idênticas, mas não necessitam de atualizar o identificador do objeto.

Para o caso de *queries* que tenham como resultado objetos, ou listas de objetos diretamente relacionados com as entidades do modelo de dados, foi criada a classe DBObjectCreator que oferece métodos para transformação do tuplo corrente de um cursor para o respetivo objeto manipulável em memória.

As *queries* são implementadas em três classes distintas: TimeMachineContainerQueries, TimeMachineClusteringQueries e TimeMachineAnalyzeQueries. A primeira contém as *queries* relacionadas com os modelos de utilização do tempo, a segunda com o processo de *clustering* e a terceira com a análise de dados para efeitos de visualização.

A classe TimeMachineSqliteDatabase herda da classe oferecida pela plataforma Android SQLiteOpenHelper que permite a criação e ligação a bases de dados SQLite. Sempre que é instanciada, é chamado o construtor de superclasse com o nome da base de dados e respetivo número de versão sendo a base de dados criada se não existir ou se o número de versão for superior ao existente. Os métodos `getWritableDatabase()` e `getReadableDatabase()` são implementados na superclasse, devolvendo uma referência para um objeto do tipo SQLiteDatabase que possibilita a interação com a base de dados através de operações CRUD ou de leitura. Uma visão geral do pacote db, que contém todas as classes que manipulam o modelo relacional, é apresentada na figura 4.6.

4.1.4 Captura e Processamento de Informação Geográfica

A captura de localização é implementada pela classe LocationService. A classe corresponde a um componente Service estendendo esta classe. Implementa ainda a interface LocationListener e GpsStatus.Listener para se poder registar como *listener* dos fornecedores de localização do sistema e do estado do GPS.

O registo do *listener* de localização é efetuado através da invocação do método

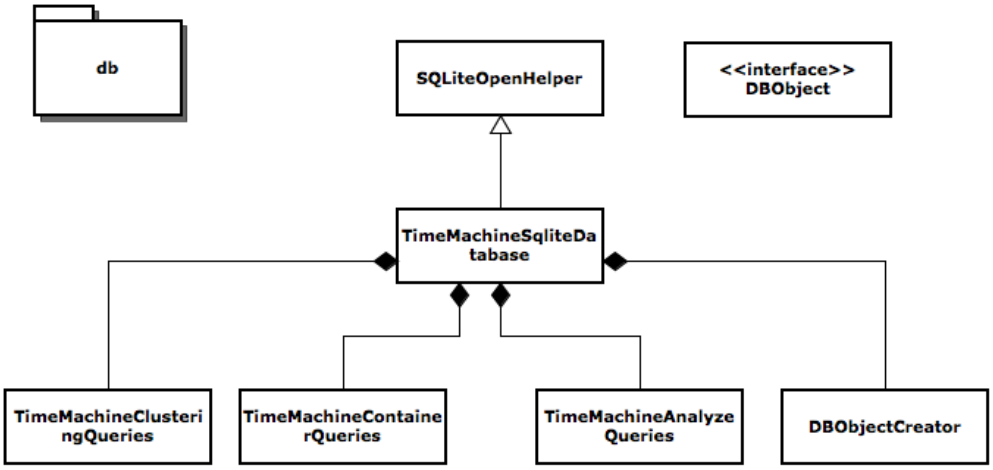


Figura 4.6: Interação objeto-relacional, pacote db

`addLocationListeners()`. Neste método é obtida uma referência para a classe LocationManager que funciona como Singleton de acesso aos serviços de localização do sistema. Esta classe oferece o método `requestLocationUpdates(String provider, long mintime, float mindistance, LocationListener listener)` que permite o registo de um *listener* para um fornecedor de localização com os parâmetros de distância e tempo mínimos para atualização. Os valores utilizados no registo são 1 minuto e 5 metros. Assim sempre que passe 1 minuto e a nova localização diste 5 metros da anterior o objeto da classe LocationService é notificada através da invocação do método `onLocationChanged(Location location)` com a nova coordenada a ser passada como parâmetro. Estes parâmetros são muito relevantes no desempenho energético da aplicação, mas também na qualidade e cobertura dos dados capturados. Parâmetros mais elevados permitem uma maior poupança da bateria e menos implicações na utilização normal do telemóvel, mas o seu relaxamento leva a descontinuidades do *tracking* do utilizador. Assim foi decidido que os parâmetros utilizados seriam bastante agressivos dado que é fundamental para a aplicação o *tracking* contínuo do utilizador ficando em aberto futuras otimizações energéticas.

Na plataforma Android os componentes Service podem correr indefinidamente se forem lançados por outro componente (ver 4.1.1). São especificamente utilizados para desenvolver operações em *background* e ainda que não corram automaticamente na sua própria *thread* podem por si próprios lançar novas para executar as operações pretendidas. Existe no entanto uma característica associada a estes componentes. O sistema operativo pode a qualquer momento, por falta de memória ou bateria, terminar o componente, e não existe nenhum mecanismo que permita explicitamente evitá-lo. A única forma de controlar esta situação é através da definição das prioridades associadas ao componente, utilizadas na heurística que determina qual deve ser terminado caso um destes

eventos se verifique. Dado que o componente de captura de localização é indispensável para a aplicação Time Machine, é necessário utilizar os mecanismos oferecidos pelo sistema para reduzir a probabilidade do mesmo ser terminado. Para tal o Service deve ser transformado de um componente *background*, sem interação com o utilizador, para *foreground* em que é visível. Isto é efetuado através do método `startForeground(int id, Notification notification)`, que tem como parâmetro um objeto correspondente a uma notificação a ser apresentada na barra de sistema. Enquanto o componente estiver a correr a notificação é apresentada na respetiva barra informando visualmente o utilizador da sua atividade. Sendo o Service algo de que o utilizador tem conhecimento, o sistema evitará terminá-lo em situações de pouca memória, optando primeiro por componentes *background* e em caso de não o poder evitar garante que o lançará de novo logo que seja possível.

Cabe ainda detalhar o momento e processo de lançamento do componente LocationService. Depois da instalação da aplicação existem duas formas do componente de captura de localização ser lançado: (1) através do lançamento da aplicação e (2) no processo de arranque do sistema. A primeira é essencialmente utilizada no momento imediatamente posterior à instalação da aplicação, sendo o objeto LocationService criado e lançado pela Activity TimeMachine (ponto de entrada na aplicação). O lançamento do componente de captura no arranque do sistema evita que o utilizador depois de reiniciar o telemóvel tenha de explicitamente aceder à aplicação para dar inicio à captura. Esta funcionalidade é oferecida através da classe LocationServiceLauncher que herda de BroadcastReceiver. Esta classe corresponde a um componente BroadcastReceiver e responde ao evento BOOT_COMPLETED registado no ficheiro Manifest.xml. Quando o sistema difunde o evento, cria também uma instância desta classe e invoca o método `onReceive(Context ctx,Intent intent)` herdado da superclasse BroadcastReceiver. O método é redefinido pela classe LocationServiceLauncher para lançar o componente LocationService, iniciando imediatamente a captura de dados de localização.

Quando o sistema notifica uma nova localização, o método `onLocationChanged(Location location)` é invocado. Este método, partindo do objeto que lhe é passado, calcula o *geohash* associado à coordenada e constrói um objeto do tipo AndroidCoordinate que corresponde à entidade Coordinate do modelo de dados. Periodicamente são escritas as coordenadas acumuladas no sistema de ficheiros. Para além disso um novo Service, denominado ClusteringService, é lançado para iniciar o processamento da informação geográfica e a extração de informação baseada na coordenada capturada.

4.1.4.1 Processamento de Informação Geográfica

O Service ClusteringService, lançado pelo método `onLocationChanged(Location location)` aquando da receção de uma nova coordenada, começa por obter uma referência para a instância do Singleton ClusteringManager. Esta classe implementa todo

o processamento da informação geográfica e de extração de locais com significado, podendo ainda lançar o processo de extração de informação sobre a rotina do utilizador. Depois de obtida a instância de ClusteringManager o método `addCoordinate(Android Coordinate toAdd)` é invocado passando em argumento a coordenada acabada de capturar. Este método começa por registar a coordenada na base de dados relacional, na tabela Coordinate, executando de seguida o algoritmo de extração de Staypoints. Este algoritmo foi adaptado daquele apresentado em 2.3, para funcionar de forma incremental e é detalhado seguidamente.

O algoritmo de extração de Staypoints (pontos de estadia) requer a manutenção de uma lista de coordenadas acumuladas como membro de dados da classe ClusteringManager. No caso da lista estar vazia então a única ação realizada é adicionar a coordenada em argumento à cabeça. Se a lista é não vazia, o algoritmo calcula uma aproximação da distância entre a coordenada capturada e a coordenada presente na cabeça da lista. Seguidamente duas verificações são efetuadas. A primeira é se a precisão da nova coordenada é aceitável e a distância é superior a um valor definido. A segunda se não existindo a precisão mínima, a distância é maior que a precisão da coordenada à cabeça da lista. Caso nenhum destes critérios seja satisfeito a lista de coordenadas acumulada é limpa e é adicionada à cabeça a nova coordenada, correspondendo esta situação a movimento do utilizador. Caso contrário é calculado o tempo decorrido entre a coordenada à cabeça e à cauda da lista e caso o valor encontrado seja igual ou superior ao mínimo tempo de estadia definido, é criado um novo Staypoint. O novo Staypoint é criado com o tempo de início correspondente ao tempo de captura da coordenada à cabeça da lista e tempo final correspondente à captura da nova coordenada detetada. A coordenada do Staypoint corresponde à coordenada média de todas as coordenadas presentes na lista no momento da sua criação. O Staypoint é inserido na base de dados relacional na tabela Staypoint e é adicionado um tuplo por coordenada presente na lista na tabela Staypoint_Coordinate correspondente à associação entre o Staypoint e cada coordenada. A lista é então limpa e a nova coordenada adicionada à cabeça. O método `addCoordinate(AndroidCoordinate toAdd)` que implementa o algoritmo descrito é apresentado em 4.4.

Listing 4.4: Algoritmo incremental de extração de Staypoints

```
1  public class ClusteringManager {
2  private List<AndroidCoordinate> coordinates;
3  (...)
4  public void addCoordinate(AndroidCoordinate toAdd) {
5        insertDBObject(toAdd);
6        if (coordinates.isEmpty()){
7            coordinates.add(toAdd);return;
8        }
9        else{
10         double distance = toAdd.distance(coordinates.get(0));
11         boolean accuracyanddistance =
```

```
12                  (toAdd.getAccuracy() <= ClusteringOptions.MINACCURACY) &&
13                  (distance > ClusteringOptions.DISTTRESH);
14          boolean noaccuracybutdistancebiggerthanaccuracy =
15                  (toAdd.getAccuracy() > ClusteringOptions.MINACCURACY)  &&
16                  (distance > toAdd.getAccuracy());
17          if (accuracyanddistance || noaccuracybutdistancebiggerthanaccuracy) {
18            int staytime = calcTimeBetween(coordinates.getFirst(),
19                           toAdd)
20            if (time >= ClusteringOptions.TIMETRESHMIN) {
21              StayPoint s = new StayPoint(computeMeanCoordinate(coordinates),
22                              coordinates.getFirst().getTime(),
23                              toAdd.getTime());
24              insertDBObject(s);
25              insertStayPointCoordinateRelations(s,coordinates)
26              addStayPoint(s);
27            }
28          coordinates.clear();coordinates.add(toAdd);
29        }
30      }
31    }
32  (...)
```

Os parâmetros TIMETRESHMIN, DISTTRESH e MINACCURACY utilizados são respetivamente 3 minutos, 3 e 90 metros. Os parâmetros foram estudados em [dB11] como sendo os adequados para a aplicação. No caso em que a execução do método dá origem à criação de um novo Staypoint, o fluxo de extração de informação prossegue através da invocação do método `addStayPoint(Staypoint s)` ainda da classe ClusteringManager. O restante processo é detalhado na secção seguinte do documento.

4.1.5 Extração de Informação

A extração de informação efetuada pela aplicação divide-se em dois blocos principais, (1) a extração de locais com significado e (2) de padrões temporais do utilizador. Os algoritmos e classes desenvolvidos para executar esta extração de informação são apresentados na presente secção.

4.1.5.1 Extração de Locais com Significado

O processo de extração de locais com significado para o utilizador é executado sempre que um novo Staypoint é detetado. Como referido anteriormente é iniciado no método `addCoordinate(AndroidCoordinate toAdd)` da classe ClusteringManager através da invocação do método `addStayPoint(StayPoint s)` com o novo Staypoint como argumento.

O método `addStayPoint(StayPoint s)` é responsável pela execução do algoritmo de extração de locais com significado para o utilizador, algoritmo esse baseado na aproximação *clustering-based* apresentada em 2.4. O método começa por obter a lista

de todos os locais com significado que possuem um Staypoint que diste menos que um certo limite do novo Staypoint. De seguida conforme o número de locais resultante, o algoritmo procede a uma de três operações possíveis: adicionar o Staypoint a um local existente, criar um novo local ou fundir locais existentes. Termina notificando o Singleton ContainerManager para a execução das operações relacionadas com a extração de padrões temporais descrita em 4.1.5.2. O método é apresentado em 4.5.

Listing 4.5: Algoritmo incremental de extração de locais com significado para o utilizador

```
List<MeaningfulLocation> owners = getPotencialOwners(s);
    int nOwners = owners.size();
    boolean newlocation = false;
    switch (nOwners) {
      case 0:
        buildLocationFromStayPointsWithoutLocation(s);newlocation = true;break;
      case 1:
        addStayPointToLocation(owners.get(0), s);break;
      default:
        merge(s, owners);break;
    }
    ContainerManager.getInstance(ctx).notifyStayPointDetected(s,newlocation);
```

Como se pode verificar o método limita-se a decidir qual a operação a executar conforme o número de possíveis locais a que o Staypoint pode ser atribuído. Interessa por isso detalhar os quatro métodos invocados que em conjunto implementam de facto o processo de *density clustering* dos Staypoints detetados.

Método getPotencialOwners(StayPoint s) Este método é responsável por obter a lista de locais a que o novo Staypoint pertence. O critério utilizado para decidir se um Staypoint pertence a um local, é possuir um Staypoint associado que esteja a uma distância deste inferior a um valor eps definido. Para tal é executada uma *query* que filtra os Staypoints por *geohash* igual ao do Staypoint a testar, efetuando posteriormente a junção com a tabela MeaningfulLocation para obtenção dos locais e Staypoints associados. Esta *query* permite uma filtragem geográfica dos dados originando uma junção de menos tuplos, otimizada pela existência de um índice no atributo geohashs da tabela Staypoint. O facto do SGBD SQLLite, versão da plataforma Android, não oferecer índices multidimensionais nem funções que permitam o cálculo de uma distância geográfica na própria *query*, levou ao desenvolvimento desta solução para evitar carregar todos os Staypoints e MeaningfulLocations para memória. Já em memória a lista de Staypoints obtidos é testada através da distância de Haversine de forma mais precisa. Todos os locais que possuam um Staypoint dentro do critério são adicionados a uma lista que é depois devolvida como resultado do método. O método é apresentado em 4.6 e o valor eps utilizado são 30 metros como sugerido em [dB11].

Listing 4.6: Método getPotencialOwners(StayPoint s)

```
List<MeaningfulLocation> getPotencialOwners(StayPoint newStayPoint) {
  List<MeaningfulLocation> result =
    getMeaningfulLocationsCloseTo(newStayPoint.getCoord());
  Iterator<MeaningfulLocation> it = result.iterator();
  while(it.hasNext()){
    MeaningfulLocation l = it.next();
    if(!isCoordinateWithin(newStayPoint.getCoord(), l))
      it.remove();
    }
  result = stabilityFilter(result,newStayPoint);
  return result;
  }
```

Para contornar o problema referido em [MGP10] de locais que crescem indefinidamente com o tempo neste tipo de aproximações, foi introduzido um filtro de estabilidade no algoritmo. A filtragem é executada pelo método `stabilityFilter`. O filtro é baseado no conceito de estabilidade de um local e, dada uma lista de locais que estão na vizinhança do novo Staypoint, procede à análise de quais destes possuem uma duração ou frequência de acesso superior a STABILITYDURATION e STABILITYFREQ, limites definidos estaticamente. Se existir pelo menos um local estável, o filtro exclui todos os locais da lista com exceção do estável mais próximo geograficamente do novo Staypoint. Impede assim a existência de uma fusão entre locais limitando o seu crescimento. Os valores utilizados para STABILITYDURATION e STABILITYFREQ são 720 minutos e 15 acessos respetivamente, e foram escolhidos através de experimentação com alguns conjuntos de dados.

Método buildLocationFromStayPointsWithoutLocation(StayPoint s) O caso em que a lista de locais com significado resultado do método anterior é vazia, significa que o Staypoint não pertence a nenhum local existente. O presente método tenta então a criação de um novo local a partir de todos os Staypoints presentes na base de dados que não pertençam a nenhum local e possuam o mesmo *geohash* do novo Staypoint. Percorrendo a lista resultante, calcula a distância de Haversine entre o novo Staypoint e cada elemento. Caso a distância seja inferior ao limite eps, incrementa a duração e frequência total acumuladas. No final do ciclo se a frequência total for superior a MINFREQ ou a duração total superior a MINDRT cria um novo local com os Staypoints que respeitam o critério de distância. O valor utilizado para MINFREQ E MINDRT vem das experiências efetuadas em [dB11] e é de 3 para MINFREQ e 10 minutos para MINDRT. O algoritmo é apresentado em 4.7.

Listing 4.7: Método buildLocationFromStayPointsWithoutLocation(StayPoint s)

```
private void buildLocationFromStayPointsWithoutLocation(StayPoint s) {
  double duration = 0;
  List<StayPoint> staypointsWithoutMeaningfulLocation =
    getStaypointsWithoutMeaningfulLocationCloseTo(s.getCoord());
```

```
5     Iterator<StayPoint> it = staypointsWithoutMeaningfulLocation.iterator();
6     while (it.hasNext()) {
7       StayPoint sp = it.next();
8       if (s.getCoord().distHaversine(sp.getCoord())<=ClusteringOptions.EPS){
9         duration += sp.getStayDuration();
10      }
11      else{
12        it.remove();
13      }
14    }
15    if(staypointsWithoutMeaningfulLocation.size()>=ClusteringOptions.MINFREQ
16      ||
17      duration >= ClusteringOptions.MINDRT){
18      MeaningfulLocation l =
19        new MeaningfulLocation(staypointsWithoutMeaningfulLocation);
20      insertDBObject(l);
21      addStayPointsToMeaningfulLocation(l,
22                        staypointsWithoutMeaningfulLocation);
23    }
24  }
```

O método insere o novo local na tabela MeaningfulLocation bem como um tuplo contendo o identificador do local e do Staypoint na tabela StaypointMeaningfulLocation por cada Staypoint associado ao novo local criado.

Método addStayPointToLocation(MeaningfulLocation owner,StayPoint s) Se a lista de locais resultante do método `getPotencialOwners(StayPoint s)` for de tamanho um, o presente método é invocado executando apenas uma operação, a inserção de um tuplo contendo o identificador do Staypoint e da MeaningfulLocation contida na lista, na tabela Staypoint_MeaningfulLocation. Fica assim associado o Staypoint ao local com significado correspondente.

Método merge(StayPoint s, List<MeaningfulLocation> owners) Caso a lista resultante de `getPotencialOwners(StayPoint s)` contenha mais de um local com significado é necessário efetuar a fusão dos mesmos. Para tal três operações têm de ser efetuadas: criar e inserir na base de dados um novo local que corresponderá ao resultado da fusão, associar os Staypoints e registos de acesso dos locais fundidos ao novo local e finalmente remover os locais fundidos da base de dados. A implementação do método é apresentada em 4.8.

Listing 4.8: Método merge(StayPoint s List<MeaningfulLocation> owners)

```
1  public void merge(StayPoint newStayPoint,List<MeaningfulLocation> toMerge){
2    MeaningfulLocation newlocation = new MeaningfulLocation(newStayPoint);
3    insertDBObject(newlocation);
4    addStayPointToMeaningfulLocation(newlocation, newStayPoint);
5    for (MeaningfulLocation toBeMerged : toMerge) {
6      updateDBObject(new MeaningfulLocation2AcessRelation(
```

```
7                                    newlocation.getId(),
8                                    toBeMerged.getId())
9                                    );
10      ContainerManager.getInstance(ctx).updateIndividualLocationStats(
11                                   newlocation.getId(),
12                                   toBeMerged.getId()
13                                   );
14      List<StayPoint> list = getMeaningfulLocationStayPoints(
15                                     toBeMerged.getId()
16                                     );
17      for (StayPoint sp : list) {
18        updateDBObject(new MeaningfulLocationStayPointRelation(
19                                       newlocation.getId(),
20                                       sp.getId())
21                                       );
22        newlocation.addStayPoint(sp);
23      }
24      removeDBObject(toBeMerged);
25      updateDBObject(newlocation);
26      }
27  }
```

Sempre que a um local com significado é adicionado um Staypoint, é necessário incrementar o atributo `nsps`, atualizar os valores médios da latitude e longitude, e o *geohash* do local. Isto é executado eficientemente tirando partido do método de cálculo da média e do armazenamento do atributo `nsps`, evitando assim a contagem do número de Staypoints associados à MeaningfulLocation e da soma das latitudes e longitudes de todos eles. Também o atributo `label` do local é atualizado contactando um serviço web de *reverse geocoding* desde que exista conectividade de rede no momento. As operações de atualização de MeaningfulLocation são muito frequentes dado que é expectável que o utilizador frequente muitas vezes os mesmos locais, pelo que o processo de atualização tem de ser o mais eficiente possível.

4.1.5.2 Extração de Padrões Temporais do Utilizador

A extração de padrões temporais do utilizador é efetuada através da construção de modelos de utilização do tempo e sua análise. O modelo desenvolvido materializa o conceito de *time containers* apresentado em [Per11] e explicado em 2.2.2 através da definição dos perfis, períodos de treino, conjunto de atributos a recolher, extração da casa do utilizador e eventos de transição entre modelos de utilização do tempo. O modelo de utilização do tempo assenta em duas componentes principais, a análise dos acessos a locais com significado e a análise ao bloco temporal dia. As duas componentes manifestam-se no modelo de dados já apresentado através das tabelas DayInContainer (4.1.3.7) e MeaningfulLocationAcessInContainer (4.1.3.6) associadas aos modelos de utilização do tempo presentes na tabela Container (4.1.3.4). De seguida, apresenta-se o processo de extração e registo

dos dados de suporte aos dois componentes referidos e também a definição dos perfis, períodos de treino, heurística de deteção da casa e eventos de transição que concretizam em conjunto a extração de padrões temporais e rotina a que o utilizador está sujeito.

Como referido anteriormente, no final da execução do método `addStayPoint(StayPoint s)`, responsável pela extração dos locais com significado para o utilizador, é invocado o método `notifyStayPointDetected(StayPoint s, boolean newlocation)` sobre a instância da classe ContainerManager para proceder à recolha dos dados de dia e acesso a local associados ao modelo de utilização do tempo corrente.

A classe ContainerManager é um Singleton que implementa o processo de recolha de dados sobre o dia e acesso a locais e gere as transições entre modelos de utilização do tempo. Mantém como membros de dados o Container, Staypoint, MeaningfulLocationAcess e DayInContainer correntes e ainda o Staypoint anterior, que atualiza sempre que invocado o método `notifyStayPointDetected(StayPoint s,boolean newlocation)`. Este método começa por atualizar o Staypoint corrente e o anterior, inserindo depois na base de dados uma associação entre o Container e StayPoint correntes juntamente com a distância em km ao Staypoint anterior. De seguida invoca os métodos `updateLocation()` para atualização do acesso a local com significado corrente, `analyzeDayChange()` para detetar o fim de um dia e termina incrementando o número de novos locais no dia se tal for indicado em argumento. O método descrito é apresentado em 4.9.

Listing 4.9: Método notifyStayPointDetected(StayPoint s boolean newlocation)

```
1  public class ContainerManager {
2      (...)
3    private Container current;
4    private StayPoint currentStayPoint;
5    private MeaningfulLocationAcess currentLocation;
6    private DayInContainer currentDayInContainer;
7    private Context ctx;
8    (...)
9    public void notifyStayPointDetected(StayPoint detected, boolean newLocation) {
10       StayPoint old = currentStayPoint;
11       currentStayPoint = detected;
12       if(currentDayInContainer == null){
13         initCurrentDayInContainer();
14       }
15       if(old==null){
16         insertDBObject(new ContainerStaypointRelation(current.getId(),
17                                 detected.getId(),0));
18       }
19       else{
20         insertDBObject(new ContainerStaypointRelation(current.getId(),
21                                 detected.getId(),
22                                     old.getCoord().
23                                 distance(currentStayPoint.getCoord())));
24       }
```

```
25        updateLocation();
26        analyzeDayChange();
27        if(newLocation)
28          currentDayInContainer.incNewPlaces();
29    }
30    (...)
```

É relevante detalhar a implementação dos métodos `updateLocation()` e `analyzeDayChange()` responsáveis diretos pela recolha dos dados associados aos acessos a locais e ao dia respetivamente.

O método `updateLocation()` começa por obter da base de dados o local com significado associado ao Staypoint corrente. Caso o local atual seja indefinido e o associado ao Staypoint atual não, então um novo acesso a local é criado e inserido na tabela MeaningfulLocationAcessInContainer associado ao Container atual. Se ambos forem indefinidos então o método retorna sem executar mais nenhuma operação. Caso o local atual não seja indefinido e o Staypoint atual esteja associado a um local, verifica se continua no mesmo ou mudou. Caso continue no mesmo atualiza os valores de fim associados ao acesso a local, caso contrário cria um novo acesso a local com os dados do Staypoint atual. Se o Staypoint não está associado a nenhum local limita-se a tornar o acesso atual indefinido. Este comportamento corresponde ao algoritmo apresentado em 4.10.

Listing 4.10: Método updateLocation()

```
1  private void updateLocation() {
2      Long idl = getStayPointLocation(currentStayPoint.getId());
3      //Not in a meaningful location
4      if(currentLocation == null){
5        //To a meaningful location
6        if(idl!=null){
7          currentLocation = new MeaningfulLocationAcess(...)
8          insertDBObject(currentLocation);
9        }
10     }
11     //In a meaningful location
12     else{
13       if(idl!=null){
14         // To a different one
15         if(idl!=currentLocation.getLocation()){
16           currentLocation = new MeaningfulLocationAcess(...)
17           insertDBObject(currentLocation);
18         }
19         // staying some more time
20         else{
21           currentLocation.setDateEnd(currentStayPoint.getTimeEnd());
22           updateDBObject(currentLocation);
23         }
24       }
25       // to a not meaningful one
26       else{
```

```
27            currentLocation = null;
28          }
29        }
30      }
```

O método `analyzeDayChange()` verifica se o Staypoint corrente pertence a um novo dia. Caso um novo dia seja detetado então recolhe da base de dados todos os acessos a locais com significado para o dia que terminou, calcula os valores para todos os atributos da tabela DayInContainer com exceção do número de novos locais (mantido em memória), e insere o objeto DayInContainer na base de dados associado ao dia que terminou e ao Container corrente.

Este método termina a recolha dos dados que formam a base do modelo de utilização do tempo. No entanto para uma definição completa é ainda necessário incluir os perfis de agregação utilizados para analisar os dados, bem como os respetivos períodos de treino e a extração do local definido como casa do utilizador. Perfis de agregação correspondem a medidas estatísticas (médias e desvios padrão) agrupadas segundo diferentes visões de tempo. Três perfis foram definidos: A, B, C procedendo-se de seguida à sua explicação.

Perfil A Sem categorização, as medidas estatísticas são agrupadas independentemente do dia em que ocorrem.

Perfil B As medidas estatísticas são agrupadas segundo duas categorias: dias de trabalho e dias de descanso (semana/fim de semana).

Perfil C As medidas estatísticas são agrupadas segundo o dia da semana: segunda, terça, . . . , domingo.

Para cada um dos perfis existem períodos de treino associados, que advém da necessidade de evitar uma contaminação de atributos como por exemplo número de locais novos por dia e número de mudanças de local por dia, durante o processo de aprendizagem inicial dos locais com significado para o utilizador. Os períodos de treino definidos são os seguintes:

Perfil A 14 dias.

Perfil B 21 dias, correspondentes a 3 semanas que cobrem 3 observações de semanas e fins de semana.

Perfil C 21 dias, correspondentes a 3 semanas que cobrem 3 observações de segundas, terças, . . . , domingos.

Na prática tendo os dados capturados para acessos a locais e dias, associados ao modelo de tempo em que acontecem, a implementação dos perfis corresponde ao cálculo de médias e desvios padrão sobre os atributos das tabelas DayInContainer e MeaningfulLocationAcessInContainer executando *slices* sobre os atributos `idcnt` (identificador do

modelo de tempo) e `weekday`. Desta forma é possível obter sempre os valores estatísticos atualizados, o que em conjunto com um mecanismo de cache por razões de eficiência oferece um sistema ágil para a modelação dos padrões temporais a que o utilizador está sujeito num determinado contexto da sua vida.

A extração do local definido como casa do utilizador é efetuada através de uma heurística muito simples. Casa é o local com significado com maior duração de estadia no Container corrente. Apesar de simples esta heurística é bastante efetiva, principalmente utilizando uma ordenação que diz respeito apenas ao Container corrente. Este facto permite à aplicação reagir a uma mudança de rotina (por exemplo por uma ida de férias) mudando a casa detetada se o Container também mudar.

Para completar a modelação dos padrões pessoais e rotina do utilizador falta apenas definir os eventos e funções de transição que dão origem a mudança entre modelos de utilização do tempo (Containers) permitindo assim a reinicialização da agregação de valores para os diferentes perfis. Pretende-se com esta funcionalidade evitar o efeito de anulação de valores se um utilizador mudar radicalmente o seu comportamento e consequentemente os padrões e rotinas a que esta sujeito.

Foi efetuado um levantamento de possíveis eventos que poderiam dar origem a transições entre Containers embora não tenha sido possível em tempo útil desenvolver um algoritmo que, sendo detetado um dos eventos definidos, elegesse o Container para onde transitar de entre os existentes ou a transição para um novo. O comportamento alternativo implementado é a transição para um novo Container sempre que um evento de transição é detetado. Um Container pode sofrer uma transição se o período de treino do perfil A tiver sido completado. De seguida apresenta-se uma lista de eventos que dão origem à transição de Container.

- A distância percorrida num dia é 5 vezes superior ao máximo percorrido num dia pelo utilizador no Container corrente.

- O utilizador foi a um local fora do *geohash* associado ao Container que não é nenhuma das regiões adjacentes (quadrículas adjacentes). O *geohash* do Container corresponde a uma quadrícula com precisão indexada à distância média percorrida pelo utilizador num dia. Quanto maior é a distância média, menor é a precisão do *geohash* e consequentemente maior é a região associada ao Container. O *geohash* utilizado corresponde à casa do utilizador.

- O utilizador não frequenta o local assinalado como casa no Container corrente durante 3 dias.

- O utilizador não frequenta nenhum dos 10 locais com maior duração de estadia no Container corrente durante 1 dia.

4.1.5.3 Predição de Comportamento do Utilizador

A predição de comportamento do utilizador implementada é baseada apenas no comportamento esperado segundo um dos perfis definidos. O valor considerado normal (médio) para um determinado atributo é aquele que a aplicação prevê que venha a acontecer. O perfil utilizado para previsão do comportamento do utilizador é o C, que permite maior diferenciação na previsão de um determinado dia (eg., hoje é domingo o comportamento previsto para o utilizador amanhã é o correspondente ao perfil C-segundas). Apesar de ser uma forma de previsão muito simplificada permite causar no utilizador a sensação de que falhando a predição é o utilizador que está fora da sua rotina e não um engano nesta.

Apesar de não ter sido implementada a técnica Cadeia de Markov descrita em 2.2.3, existiu o cuidado de orientar o modelo de dados de forma a que fosse possível no futuro. Assim para calcular as probabilidades de transição de primeira ordem (sem memória) basta obter todos os acessos a um determinado local e o correspondente tuplo seguinte efetuando de seguida a divisão do número de observações pelo número total de transições. Para a extração de Cadeias de Markov de segunda ordem (com memória do local anterior) é necessário obter o tuplo anterior e seguinte a todos os acessos para um determinado local. A implementação requer testes de desempenho visto que é expectável que seja um algoritmo que consome bastantes recursos de processamento, mas é de possível implementação com o modelo desenvolvido. É até possível calcular a Cadeia de Markov de um determinado local efetuando um *slice* nos dados de acordo com um dos perfis definidos, possivelmente aumentando a capacidade de predição pelo maior ajustamento ao contexto corrente.

4.1.6 Visualização

O desenvolvimento das visualizações foi efetuado em dois módulos principais, um para a extração de informação necessária na classe Analysis e outro para a componente visual propriamente dita através de componentes do tipo Activity. Seguidamente detalha-se a realização de cada uma das visualizações desenvolvidas segundo os dois módulos referidos.

4.1.6.1 Visualização Distância Diária Relativa

A classe DailyDistanceVizActivity estende a classe Activity e no início do seu ciclo de vida (método `onCreate()`) lança uma tarefa assíncrona que carrega a informação da distância diária através da invocação do método `getAllDaysInfo()` sobre o objeto Analysis mantido como membro de dados. O objeto Analysis requer da base de dados a informação de todos os dias registados no Container corrente que lhe é entregue como uma lista ligada de objetos do tipo DayInContainer (correspondentes à entidade DayInContainer). Na posse desta lista o objeto Analysis calcula o dia correspondente à máxima

distância percorrida para de seguida proceder à normalização de todos os valores de distância entre 0 e 1, estabelecendo assim a relação entre cada elemento e o máximo alguma vez atingido no Container. Devolve os valores normalizados associados à informação da data em que ocorreram à classe DailyDistanceVizActivity que procede de seguida ao desenho dos elementos gráficos.

O desafio encontrado no desenho desta visualização foi o facto da plataforma Android não oferecer nenhum *widget* que permita o desenho através de primitivas 2D e *scroll* por parte do utilizador para navegação numa visualização de dimensões superiores aos *canvas* visível. A criação de um um bitmap em memória de toda a visualização é incomportável visto que o número de dias a desenhar cresce linearmente com o decorrer do tempo de utilização da aplicação. Assim foi desenvolvido um *widget* que estende a classe BaseAdapter utilizada para fazer ligação entre elementos de uma lista e a sua visualização. Esta classe mantém a associação entre a posição virtual da visualização e o elemento da lista respetivo, invocando o método `View getView(int position, View convertView, ViewGroup parent)` com os argumentos respetivos. Mantendo em memoria a lista com a informação dos dias, é possível através da redefinição deste método devolver apenas o bitmap contendo o elemento gráfico correspondente à posição passada em argumento que é depois desenhada no ecrã como uma imagem pela superclasse. Sabendo que em cada momento estão visíveis em simultâneo alguns dias, mantém-se uma cache com exclusão segundo a política LRU (*Least Recently Used*) dos bitmaps a serem desenhados e daqueles que não estando visíveis foram usados recentemente. Assim o *scroll* é executado de forma suave melhorando a interação com o utilizador.

A implementação do desenho individual de cada bitmap é bastante simples. Estando os valores normalizados entre 0 e 1 a altura máxima da barra corresponde ao valor máximo de distância observado, sendo o tamanho das restantes barras calculado através de uma transformação linear do seu valor de distância normalizado. A barra é desenhada como um retângulo preenchido com uma cor que varia entre amarelo e vermelho indexado da mesma forma que o seu tamanho. Adicionalmente, é ainda desenhada uma linha horizontal com a informação do dia a que a barra diz respeito, pintada com verde se o dia é fim de semana e vermelho caso contrário.

4.1.6.2 Visualização Forma Geométrica

A visualização Forma Geométrica é implementada através da classe SlowFastVizActivity. Começa por requerer de forma assíncrona informação sobre o número de mudanças de local observadas no dia atual e também sobre o máximo número de mudanças de local observadas num dia, no contexto do modelo de utilização do tempo atual. Estes valores são obtidos através do objeto do tipo Analysis que intermedeia a camada de visualização e de dados, invocando os métodos `getDayActivity()` e `getMaxActivity()`.

O valor do máximo obtido corresponde à deformação máxima do quadrado originando um efeito visual de uma linha. Efetuando uma transformação linear do valor observado até ao momento balizado pelo limite máximo correspondente ao máximo de transições observado, calcula a deformação a aplicar ao quadrado desenhando-o de seguida com cor vermelha. Se o valor obtido for superior ao máximo registado então o quadrado não existirá sendo desenhado apenas um ecrã totalmente preto.

4.1.6.3 Visualização Qual é a Cor deste Local?

A classe PlaceColorActivity é a responsável pela implementação desta visualização. Obtém a partir da classe Analysis, de forma assíncrona, a ordenação por frequência de acesso de todos os locais importantes presentes na base de dados. De seguida verifica a que local pertence a última coordenada capturada e conforme a posição do local na ordenação, atribui de forma linear uma cor entre o amarelo e o vermelho para locais com mais frequência e menos frequência respetivamente. Se a coordenada não pertencer a nenhum local presente na lista, desenha o vermelho puro no ecrã assinalando assim o maior desconhecimento possível na escala definida.

4.1.6.4 Visualização Quadrados de Familiaridade

A visualização Quadrados de familiaridade tira partido dos perfis definidos no modelo de utilização do tempo para apresentar em conjunto os três níveis de familiaridade do local. É implementada através da classe SquareColorActivity que obtém da classe Analysys, assincronamente, a ordenação dos locais por frequência de acesso e desempate por duração de estadia, mas efetuando um *slice* nos dados segundo os perfis A, B e C. Assim são obtidas três ordenações distintas de acordo com o dia corrente. Analisando a que local pertence a última coordenada capturada, encontra a sua colocação nas três ordenações obtidas e atribui uma cor numa escala de cinzento a cada uma das três. Uma posição mais próxima do primeiro lugar na ordenação é transformada numa cor mais próxima do branco, e um lugar mais próximo do fim da lista numa cor mais próxima do preto. Para cada ordenação que não contenha o lugar atual é atribuído o preto puro indicando um comportamento nunca observado no perfil correspondente.

São então desenhados três quadrados centrados no ecrã, pintados com a cor correspondente, com o o quadrado exterior indexado ao perfil A, o intermédio ao B e o interior ao C.

4.1.6.5 visualização Pequenos Retângulos de Tempo

A classe TimeMachineColorLinesActivity é responsável pela implementação desta visualização e requer um processo de extração de informação complexo e executado em três fases distintas. A complexidade advém de não existir uma entidade no modelo de dados que represente a recolha de atributos por hora. Este facto obriga à manipulação, através de clausulas *group by*, dos acessos a locais.

A primeira fase diz respeito à obtenção da média de atividade por hora no período respeitante à semana anterior ao dia atual e aos valores de atividade observados em cada uma das vinte e quatro horas dos mesmos dias. Para tal a classe Analysis requer da base de dados uma lista de valores por hora, invocando o método `getFrequencyByHourInDay(TimeMachineDate start, TimeMachineDate end,long idcnt)` passando como argumentos a data inicial e final que delimitam o período a obter e o identificador do modelo de utilização do tempo corrente. Na base de dados é executada um *query* que efetua um *slice* nos dados por identificador de Container e procede à contagem do número de acessos a locais agrupados por dia e hora do dia. Tendo esta lista, a classe Analysis calcula a média semanal por hora que retorna à classe TimeMachineColorLinesActivity, juntamente com os valores por hora, encapsulados num objeto do tipo WeeklyHourAnalysis. A segunda fase executa o mesmo procedimento mas apenas para obter os valores por hora para a porção do dia que já passou. Termina com a terceira fase responsável pela predição do restante do dia corrente e dos dois dias seguintes. Para prever o dia d_i recolhe os valores de atividade observados por hora para o dia d_{i-7} e d_{i-14} e atribuí como valor esperado para a hora, a média das observações de ambos.

Para proceder ao desenho a classe TimeMachineColorLinesActivity divide um retângulo de tamanho predefinido em dez retângulos verticais correspondentes a sete dias passados, ao dia atual e a dois dias futuros. Cada retângulo vertical é depois dividido em vinte e quatro retângulos de igual tamanho correspondendo às vinte e quatro horas do dia. Os retângulos são pintados com uma cor numa escala de cinzento correspondendo à distância entre o valor de atividade calculado para a hora e a média dos sete dias passados. Uma cor mais clara está mais próxima do normal e uma escura de uma situação anormal no período respeitante à última semana. É ainda desenhada uma barra indicativa da hora corrente, que é atualizada de hora em hora, sendo recalculados apenas os valores do dia corrente. Quando é detetada uma mudança de dia, toda a visualização é recalculada, procedendo-se a novo processo de extração de informação ajustado ao novo período de tempo.

4.2 Simulador

O Simulador partilha grande parte das classes desenvolvidas com a aplicação móvel. Estas classes foram desenvolvidas inicialmente para o Simulador por facilidade e rapidez de desenvolvimento numa plataforma *desktop*, tendo sido posteriormente adaptadas e integradas no fluxo da aplicação móvel Time Machine. Na presente secção apresenta-se o fluxo aplicacional do Simulador e as principais diferenças em relação à aplicação móvel, com especial foco no processo de simulação e menos na modelação já apresentada na secção anterior.

Na classe TimeMachineSimulatorUI, ponto de entrada na aplicação, é executado o carregamento do ficheiro em formato CSV escolhido pelo utilizador que contém as coordenadas. Através da invocação do método `loadCooordinates(String path)` da

classe CoordinatesLoader obtém uma lista de objetos do tipo AndroidCoordinate com todas as coordenadas contidas no ficheiro ordenadas pelo momento de captura (mais antiga à cabeça). É criada então uma instância da classe Simulator passando ao construtor a lista de coordenadas carregada e um objeto do tipo SimulatorParams. Este último contém os parâmetros de simulação a utilizar, nomeadamente o intervalo de tempo entre execuções de um passo da simulação e a quantidade de tempo a avançar nos dados por passo de simulação.

A aplicação apresenta na janela a informação sobre os dados carregados e aguarda que o utilizador pressione o botão Start. Quando o evento de clique no botão Start é disparado, é criado um objeto do tipo SimulationStepTask que estende a classe TimerTask e redefine o método `void run()`, onde será executado o código correspondente a um passo de simulação. Esta tarefa é agendada num objeto Timer que controlará a sua execução segundo a periodicidade indicada em argumento. A cada invocação levada a cabo pelo Timer do método `run()`, é calculada a data e hora limite para o passo de simulação através da soma do período de tempo de simulação selecionado pelo utilizador (hora, dia, mês). O Simulador entrega à classe ClusteringManager, uma a uma, todas as coordenadas presentes na lista com data inferior ao final do período de simulação a partir da coordenada seguinte à última entregue. A classe ClusteringManager executa as operações já referidas no contexto da aplicação móvel fazendo persistir a informação extraída na base de dados relacional de suporte. É assim reproduzido o processo de captura e tratamento das coordenadas tal como se acontecesse na aplicação móvel embora de forma mais rápida, permitindo ao utilizador parar o processo a qualquer momento inspecionando o estado atual dos dados extraídos. A classe SimulationStepTask, contendo a lógica de simulação é apresentada em 4.11.

Listing 4.11: Classe SimulationStepTask

```
1  class SimulationStepTask extends TimerTask {
2      @Override
3      public void run() {
4        if (isSimulationComplete()) {
5          log.append("Simulation␣Complete\n");
6          KmlWriter writer = new KmlWriter();
7          try {
8            writer.writeLocations(TimeMachineDatabase.getMeaningfulLocations(),
9                        "locs.kml");
10         } catch (IOException e) {}
11         stopSimulation();
12         return;
13       }
14       DateTime limit = null;
15       String mode = "";
16       switch (params.getStp()) {
17       case HOUR:
18         limit = currentDate.plusHours(1);mode = "+␣1␣Hour";break;
19       case DAY:
```

```
20          limit = currentDate.plusDays(1);mode = "+␣1␣Day";break;
21        case MONTH:
22          limit = currentDate.plusMonths(1);mode = "+␣1␣Month";break;
23        }
24        log.setText("Simulation␣Step␣:␣" + simulationstep + "␣(" + mode
25            + ")␣until␣" + new TimeMachineDate(limit).toStringDateTime()
26            + "\n");
27        while (iterator.hasNext() &&
28            currentEvent.getTime().getDate().isBefore(limit)) {
29            ClusteringManager.getInstance().addCoordinate(currentEvent);
30          currentEvent = iterator.next();
31        }
32        currentDate = limit;
33        simulationstep++;
34      }
35    }
```

4.3 Considerações Finais

A realização da aplicação móvel Time Machine revelou-se uma tarefa com desafios em várias áreas, situação que possibilitou o desenvolvimento de algumas soluções que podem ser consideradas bastante relevantes.

As limitações impostas pelas características inerentes a um dispositivo de computação móvel, como a capacidade de processamento e memória inferior a um computador pessoal, juntamente com as limitações impostas pelo sistema SQLite, obrigaram a uma disciplina na manipulação dos dados que conduziu à integração de técnicas que revelaram funcionar de forma muito positiva. Interessa assinalar particularmente a utilização do sistema Geohash (4.1.3.8) como forma de otimização das frequentes *queries* de natureza geográfica, reduzindo o conjunto de dados a analisar no processo de extração de informação e permitindo a criação de índices sobre os atributos geográficos.

A extração de locais com significado para o utilizador, através de técnicas de *density clustering* é alvo de diversas publicações como referido no trabalho relacionado. No entanto, a utilização deste tipo de técnicas em tempo-real, com a deteção dos locais a ser efetuada no próprio dispositivo móvel à medida que o *tracking* da localização é efetuado é inovadora e revelou-se surpreendentemente viável e eficaz. A adaptação destas técnicas, usualmente efetuadas em *batch* e *offline*, ao ambiente móvel são detalhadas em 4.1.5.1 e permitiram a construção de um modelo que captasse os padrões temporais a que o utilizador está sujeito.

O trabalho realizado com vista a capturar os padrões temporais do utilizador partiu de uma base teórica pouco desenvolvida, como se pode verificar pelas poucas referências nesta área. O trabalho efetuado teve de se concentrar no desenvolvimento teórico da modelação dos padrões temporais do utilizador e ao mesmo tempo na sua integração num sistema computacional. O trabalho efetuado em ambas as componentes é descrito

em 4.1.5.2 e apesar de não se poder considerar completo, constitui uma base sólida para futuro desenvolvimento.

Foram ainda desenvolvidas um conjunto de visualizações descritas em 4.1.6, que apesar de não explorarem todas as capacidade da aplicação desenvolvida, permitiram a apresentação visual de alguns resultados que indiciam a potencialidade do sistema.

5

Avaliação

Neste capítulo é apresentado o estudo levado a cabo no sentido de proceder a uma avaliação sobre a utilização da aplicação móvel e de cada um dos componentes visuais que a integram. O estudo foi realizado através de inquéritos não presenciais a um grupo de utilizadores da aplicação, após utilização durante um período de pelo menos duas semanas.

5.1 Caracterização do Grupo Inquirido

O grupo inquirido é constituído por 7 elementos, 5 do género masculino e 2 do feminino. A maior concentração etária situa-se na faixa dos 20 a 30 anos com 4 elementos, existindo 2 elementos com idade superior a 40 anos (44 e 53) e 1 elemento com 14 anos. Todo o conjunto inquirido se considera familiarizado com tecnologias informáticas e usa diariamente computadores, telemóveis e a Internet. Aproximadamente 71% dos inquiridos considera o telemóvel totalmente presente no seu dia a dia, já os restantes consideram-no medianamente presente.

Em relação ao estilo de vida, numa escala de 1 (menos) a 5 (mais), aproximadamente 57% classifica o seu nível de atividade com 4, 28.5% com 3 e 14.3% com 2. Em relação ao nível de rotina, na mesma escala, 43% classifica o seu nível de rotina com 5, 29% com 4, 14% com 3 e 14% com 2. Apesar da tendência para um nível de rotina e atividade mais elevados o conjunto de inquiridos abrange diferentes níveis nos dois atributos. Aproximadamente 85% indica ter horários regulares, 71% possui um local de trabalho fixo e afirma conseguir definir categoricamente um local como sendo a sua casa.

Apesar do conjunto avaliador não ser da dimensão necessária para conclusões definitivas, permite a recolha de algumas indicações sobre o funcionamento da aplicação que

podem ser bastante relevantes para o prosseguimento do seu desenvolvimento.

5.2 Análise de Resultados

Caracterizado o grupo de avaliação cabe agora analisar os resultados obtidos. Esta análise será efetuada segundo as secções presentes no inquérito de avaliação (ver anexo 7) que cobrem as várias funcionalidades da aplicação.

5.2.1 Utilização Geral

Para recolher *feedback* sobre a utilização geral da aplicação os utilizadores inquiridos começaram por escolher três adjetivos para classificar a sua experiência. Os resultados podem ser observados na figura 5.1. A maioria dos adjetivos escolhidos possuem uma conotação positiva tendo em conta os objetivos da aplicação. Aproximadamente 33% dos utilizadores consideraram a experiência interessante e 19% inovadora, o que reafirma a importância do conceito subjacente à aplicação. No entanto 10% referem que a experiência é complexa o que pode estar relacionado com o pouco desenvolvimento efetuado no que respeita à interface com o utilizador.

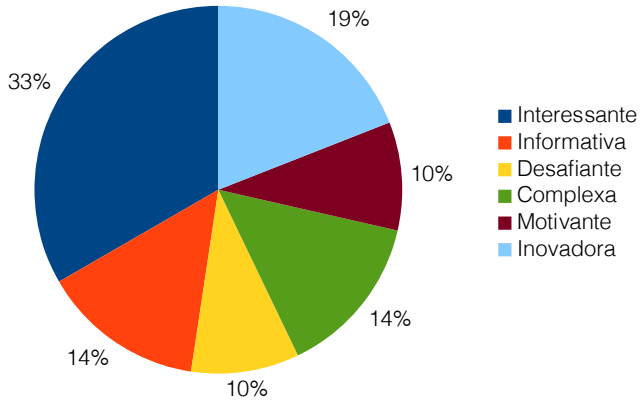


Figura 5.1: Resultados da avaliação: experiência geral de utilização.

Em relação à identificação dos locais com significado para o utilizador os resultados são muito positivos. Numa escala de 1 (incompleta) a 5 (completa) todos os utilizadores classificaram esta funcionalidade com um valor de 4 e 5 (5 - 57%, 4 - 43%) como pode ser verificado na figura 5.2. Estes resultados indiciam que a identificação de locais com significado implementada forma uma base sólida para a extração dos padrões temporais do utilizador.

Os utilizadores foram também inquiridos acerca do impacto que a aplicação tem na utilização geral do telemóvel. A tarefa de *tracking* da localização do utilizador requer um consumo energético assinalável e é importante aferir de que forma se traduz na prática

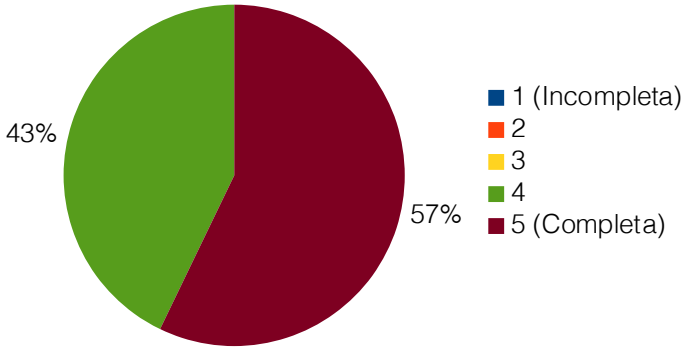


Figura 5.2: Resultados da avaliação: identificação de locais com significado.

esta necessidade. Os utilizadores apontaram um impacto bastante elevado da aplicação na utilização do telemóvel, numa escala de 1 (pouco) a 5 (muito), 43% indicaram um impacto de 4, 29% de 5 e 29% de 3. Estes dados, apresentados na figura 5.3, indiciam que alguma coisa deve ser feita para diminuir este impacto, visto que os utilizadores podem desistir de utilizar a aplicação se esta for muito prejudicial à normal utilização do seu dispositivo do qual dependem para a realização de tarefas mais importantes como a comunicação telefónica ou textual.

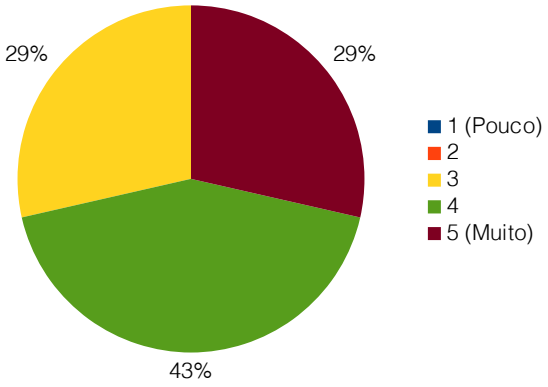


Figura 5.3: Resultados da avaliação: impacto da aplicação na utilização do telemóvel.

No que respeita à eficácia da transmissão dos conceitos associados à rotina, numa escala de 1 (ineficaz) a 5 (eficaz), 57% dos inquiridos consideram que a aplicação o faz com uma eficácia de 4. Os restantes classificam a eficácia com um valor de 3. São valores que podem ser considerados positivos visto que a aplicação apresenta as visualizações de forma isolada sem um modelo de navegação elaborado e com uma interface pouco desenvolvida. Os resultados são apresentados na figura 5.4.

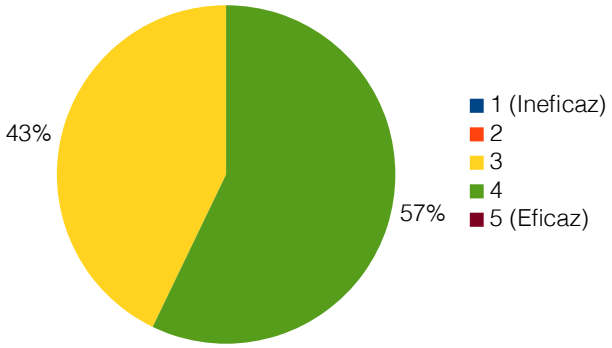


Figura 5.4: Resultados da avaliação: eficácia na transmissão de rotina pelo conjunto de visualizações.

5.2.2 Visualizações

Os utilizadores escolheram um conjunto de três adjetivos para classificação da experiência de utilização de cada visualização e responderam a um conjunto de perguntas específicas relacionadas com estas. De seguida analisam-se os resultados obtidos para cada uma.

5.2.2.1 Visualização Distância Diária Relativa

Quando inquiridos sobre a experiência de utilização da visualização Distância Diária Relativa os utilizadores dividem-se bastante na sua opinião. Os resultados obtidos podem ser observados na figura 5.5. A classificação que se destaca é informativa com 29% seguida de interessante e simples com 14%. De referir ainda que 10% dos utilizadores consideram a visualização confusa. Observando a figura 5.6 pode verificar-se, numa escala

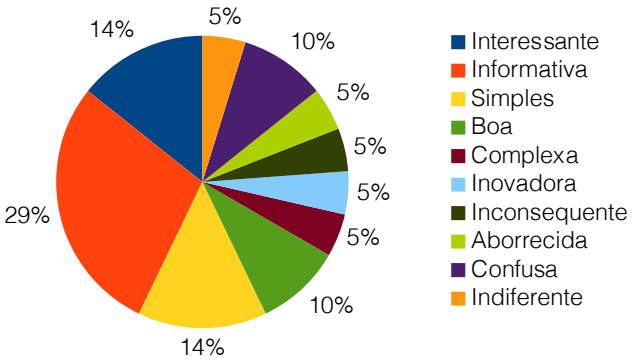


Figura 5.5: Resultados da avaliação: visualização Distância Diária Relativa, experiência de utilização.

de 1 (não transmite) a 5 (transmite), que a maioria dos utilizadores (86%) classifica com 4 ou 5 a capacidade da visualização transmitir as relações entre as distâncias percorridas

por dia, o objetivo principal da visualização. Quanto à capacidade de transmissão da quantidade de distância percorrida por dia os resultados, na mesma escala, são de 29%, 43%, 29% para as classificações 3, 4, 5 respetivamente.

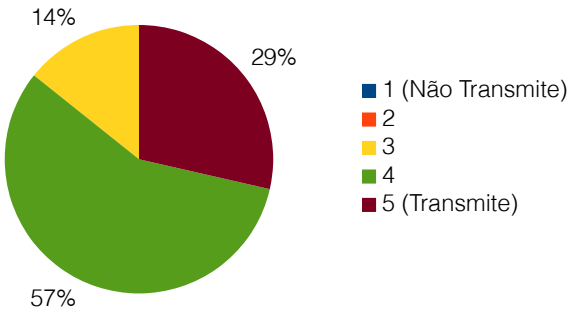


Figura 5.6: Resultados da avaliação: visualização Distância Diária Relativa, transmissão das relações entre distâncias diárias.

No que respeita à fluidez de navegação ao longo dos dias o *feedback* dos inquiridos é positivo com 86% dos utilizadores a classificarem, numa escala de 1 (não fluido) a 5 (fluido), com 4 (57%) ou 5 (29%) e os restantes com 3.

5.2.2.2 Visualização Forma Geométrica

Os utilizadores consideram a experiência de utilização da visualização Forma Geométrica inovadora (29%), desafiante (24%) e divertida (19%). Todos os adjetivos escolhidos pelos utilizadores possuem uma conotação positiva no contexto da visualização e indicam uma relação de empatia e prazer de utilização.

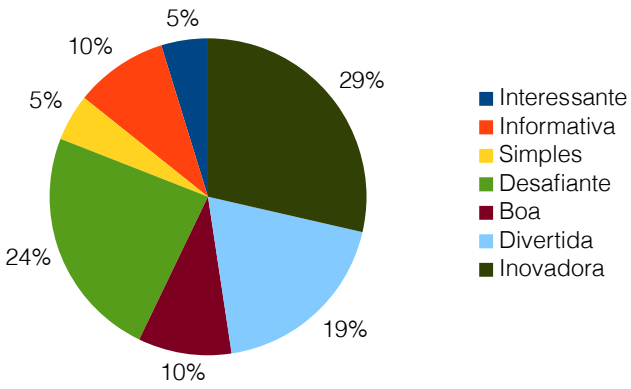


Figura 5.7: Resultados da avaliação: visualização Forma Geométrica, experiência de utilização.

A visualização provoca uma reação significativa dos utilizadores inquiridos dado que

numa escala de 1 (nada desafiante) a 5 (totalmente desafiante), 57% classificam a visualização com 5 e os restantes com 4, no que diz respeito ao desafio que sentem para deformar o quadrado sendo mais ativos. A figura 5.8 apresenta o gráfico correspondente.

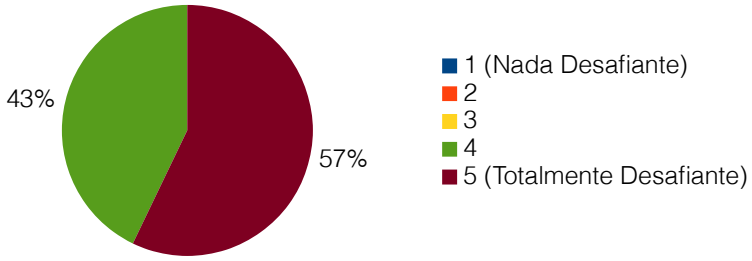


Figura 5.8: Resultados da avaliação: visualização Forma Geométrica, desafio.

No que respeita à capacidade da visualização transmitir o nível de atividade a que estão sujeitos, os utilizadores classificam a visualização com 4 (57%) e 5 (43%) numa escala de 1 (não transmite) a 5 (transmite).

5.2.2.3 Visualização Qual é a Cor deste Local?

A maioria dos inquiridos relatam a sua experiência de utilização da visualização Qual é a cor deste local? como simples (28%) e informativa (22%) como se pode observar na figura 5.9.

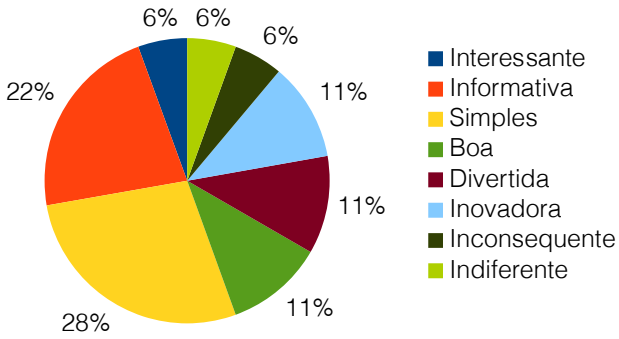


Figura 5.9: Resultados da avaliação: visualização Qual a Cor do Local?, experiência de utilização.

A nível de rapidez de atualização de acordo com o local os resultados não são tão satisfatórios. 50% classifica com 3, numa escala de 1 (lentamente) a 5 (rapidamente), a velocidade de reação da visualização ao local onde se encontra (figura 5.10). Isto pode dever-se ao facto de o utilizador percecionar mais rapidamente o tipo de local onde está por duas razões: sabe para onde se desloca e sabe se o sítio é ou não conhecido. A aplicação por sua vez tem de aguardar a captura de coordenada e analisar a que local

com significado pertence, necessitando de mais tempo para o conseguir e dando origem a algum defraudar da expetativa do utilizador.

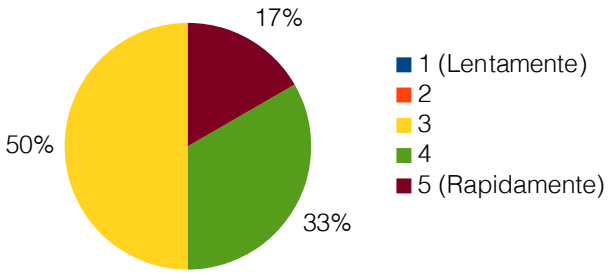


Figura 5.10: Resultados da avaliação: visualização Qual a Cor do Local?, rapidez de reação ao local onde o utilizador se encontra.

5.2.2.4 Visualização Quadrados de Familiaridade

Os resultados apurados para a visualização Quadrados de Familiaridade permitem concluir, em relação à experiência de utilização, que a maior parte dos inquiridos considera que esta é inovadora (20%), informativa (20%) e interessante (20%). No entanto alguns utilizadores classificam a experiência como confusa (7%), complexa (7%) e aborrecida (7%) o que pode estar ligado a alguma dificuldade de compreensão do seu funcionamento. A distribuição das classificações pode ser observada na figura 5.11.

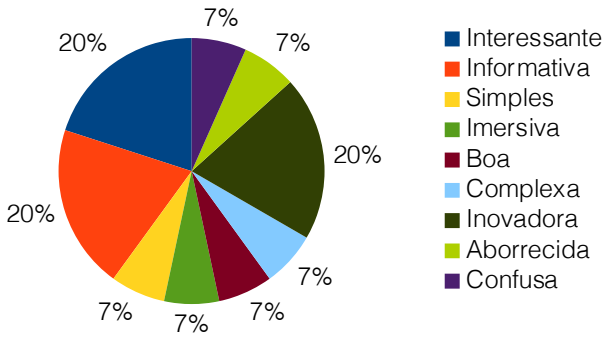


Figura 5.11: Resultados da avaliação: visualização Quadrados de Familiaridade, experiência de utilização.

Quanto à capacidade de transmissão da familiaridade no acesso ao local os inquiridos, numa escala de 1 (não transmite) a 5 (transmite), classificaram com 3 e 4 40% das vezes e com 5 20% das vezes. O gráfico correspondente é apresentado na figura 5.12. Estes resultados, em conjunto com os da experiência de utilização, indicam que é necessário algum desenvolvimento adicional no sentido de comunicar mais claramente a metáfora subjacente à visualização.

No que respeita à rapidez de atualização da visualização, numa escala de 1 (lentamente) a 5 (rapidamente), 40% dos utilizadores classificam com 3 e 40% com 5 sendo os restantes classificados com 4.

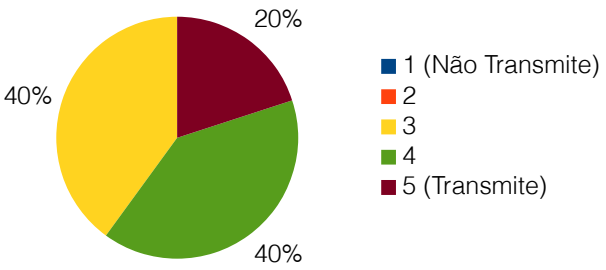


Figura 5.12: Resultados da avaliação: visualização Quadrados de Familiaridade, eficácia na transmissão da familiaridade do acesso ao local.

5.2.2.5 Visualização Pequenos Retângulos de Tempo

A experiência de utilização é descrita pela maioria dos inquiridos como interessante (33%), informativa (27%) e inovadora (20%). Nenhum dos adjetivos utilizados na classificação da experiência possui uma conotação negativa o que indicia uma resposta positiva à mesma. A distribuição das classificações utilizadas é apresentada na figura 5.13. A maioria dos utilizadores consideram ainda, como pode ser observado na figura 5.14, que a visualização transmite a diferença à rotina horária de atividade. Numa escala de 1 (não transmite) a 5 (transmite), os utilizadores classificaram a visualização com 4 e 5, 40% das vezes e com 3, 20% das vezes.

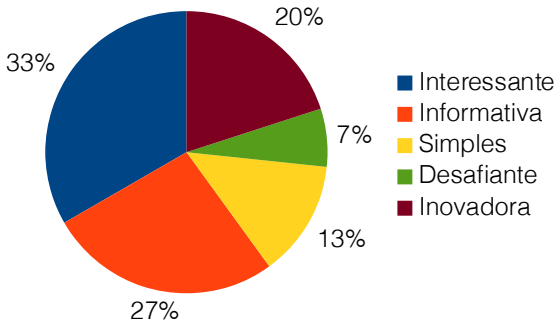


Figura 5.13: Resultados da avaliação: visualização Pequenos Retângulos de Tempo, experiência de utilização.

Em relação à funcionalidade de previsão os utilizadores estabelecem uma relação entre o previsto e a realidade, numa escala de 1 (raramente) a 5 (frequentemente), de 4 60% das vezes e de 3 e 5, 20% das vezes. Estes resultados podem ser considerados positivos

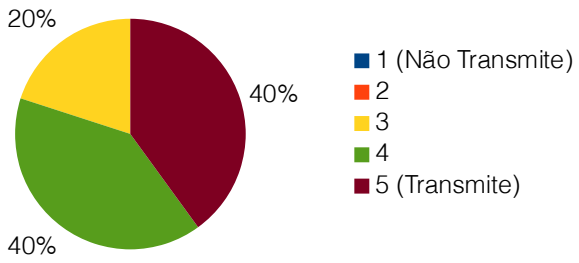


Figura 5.14: Resultados da avaliação: visualização Pequenos Retângulos de Tempo, transmissão da diferença à rotina horária de atividade.

dado o carácter experimental e complexo da funcionalidade.

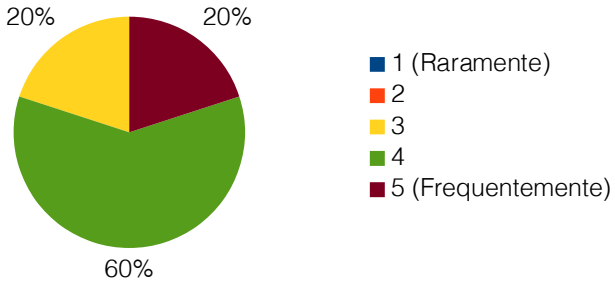


Figura 5.15: Resultados da avaliação: visualização Pequenos Retângulos de Tempo, relação entre previsão e realidade.

5.3 Considerações Finais

A avaliação efetuada permitiu a recolha de algumas indicações em relação aos vários componentes da solução desenvolvida. O conjunto de utilizadores avaliado é pequeno e por isso as conclusões a retirar devem ser contextualizadas, não podendo ser declaradas como definitivas. Só o desenvolvimento da aplicação até um estado em que permita uma utilização em grande escala permitirá a sua avaliação por um conjunto grande e diversificado de utilizadores, que conduza a conclusões mais informadas. No entanto, a opinião dos utilizadores em relação à identificação dos locais com significado é positiva assim como o é em relação à generalidade das visualizações. O impacto que a aplicação tem na utilização do telemóvel é um fator negativo apontado pela maioria dos utilizadores, devendo por isso existir uma atenção especial no desenvolvimento que permita minorar este problema.

6

Conclusões e Trabalho Futuro

O presente capítulo efetua uma análise final ao trabalho desenvolvido e aos vários componentes da solução apresentada. Apresenta ainda uma visão sobre o trabalho futuro no contexto do projeto Time Machine.

6.1 Conclusões

O objetivo principal da dissertação, o desenvolvimento de uma aplicação móvel que através do *tracking* da localização do utilizador extrai os seus padrões temporais e os apresenta visualmente, levantava um conjunto de desafios a que cumpria responder. A partir da captura das coordenadas geográficas surge a necessidade de detetar pequenas vizinhanças geográficas onde o utilizador permanece, tendo sido desenvolvido um algoritmo que à medida que as coordenadas são capturadas cria pontos de estadia contendo a coordenada média e o tempo de início e fim de estadia. Esta entidade, o ponto de estadia, é a base de um nível de análise superior: os locais com significado para o utilizador. Com base na frequência e duração de acesso, um algoritmo de *clustering* foi implementado permitindo a extração em tempo real dos locais com significado para o utilizador a partir dos pontos de estadia identificados.

Monitorizando o acesso aos locais com significado pelo utilizador, foi possível implementar um processo de recolha de atributos indexados ao dia e aos locais, que constituem um repositório de dados facilitador da tarefa de extração de padrões temporais e rotinas a que o utilizador esta sujeito. A formalização de um modelo de acesso a este repositório de dados permitiu a definição de modelos de utilização do tempo, estando o utilizador sujeito a um destes modelos em cada momento e podendo transitar para novos, através de eventos que assinalem uma diferença significativa na sua vida.

Toda esta funcionalidade começou por ser desenvolvida numa aplicação *desktop* possibilitando a simulação do processo de captura com dados recolhidos anteriormente, sendo posteriormente o código integrado na aplicação móvel.

Com o trabalho de captura, processamento e extração de informação efetuado foram desenvolvidas algumas visualizações considerando os requisitos de subjetividade e promoção de reflexão e auto-conhecimento identificados. Estas visualizações, de diferentes domínios temporais (agora/semana/histórico), permitiram uma visão contextual sobre o conhecimento obtido pela aplicação em tempo real.

A identificação dinâmica dos locais com significado para o utilizador é merecedora de uma análise individual. De uma forma geral, o algoritmo deteta todos os locais com significado para o utilizador. No entanto existe um caso de falha que importa discutir. Se o utilizador percecionar dois locais com significado próximos, o algoritmo poderá ter dificuldade em mantê-los como entidades separadas. Se os locais forem detetados num horizonte temporal distinto (a deteção de um é muito anterior ao outro) o filtro de estabilidade introduzido resolve o problema, mantendo as suas fronteiras separadas e atribuindo o ponto de estadia ao local com a coordenada média mais próxima. Por outro lado se o surgimento de ambos os locais for anterior ao alcance da estabilidade por um deles, a aplicação procede à fusão destes com consequências na modelação da rotina que podem ser graves se ambos os locais forem muito relevantes. É difícil auditar o algoritmo dado que a vizinhança é testada para cada ponto de estadia pertencente a um local com significado. Isto dá origem a uma fronteira não circular tornando por isso difícil o ajustamento do parâmetro do raio.

Do ponto de vista da otimização dos vários algoritmos relacionados com uma componente geográfica, a dissertação contribui de forma muito positiva com a integração do sistema Geohash e do SGBD SQLite. Esta integração possibilita a filtragem e *queries* de proximidade geográfica efetuadas de forma eficiente, ultrapassando a limitação da inexistência de índices multidimensionais.

A modelação da rotina é também uma das contribuições que interessa relevar. Existe uma escassez de trabalho nesta área e o desenvolvimento efetuado, a nível de modelação e implementação, pode ser considerado bastante positivo. O modelo desenvolvido é conceptualmente muito interessante e orientado a uma implementação computacional. Alguns dos resultados expressos através das visualizações, permitem concluir que o modelo é funcional, apresentando indicadores de rotina e suas quebras em relação a atributos como a atividade/inatividade e novidade/familiaridade.

6.2 Trabalho Futuro

O trabalho por desenvolver incide principalmente na interface gráfica da aplicação. Um conjunto de visualizações não constitui por si só uma aplicação que o utilizador tenha prazer em usar. Deve ser realizado um processo de conceptualização da interface, em

que devem ser integradas as visualizações por realizar. A interface deve ser uma entidade coerente, com um modelo de navegação entre os blocos visuais que faça sentido e tenha ele própria um significado para o utilizador. Para além disso devem ser procuradas formas de integração e adaptação dos protótipos visuais *Desktop* desenvolvidos na aplicação móvel. Esta tarefa deve ser efetuada após a conceptualização da interface a implementar, integrada na visão geral da mesma.

Futuramente é ainda necessário implementar a transição entre modelos de utilização do tempo de forma a permitir a reutilização dos modelos registados. Esta funcionalidade requer o desenvolvimento de um algoritmo que permita, depois da deteção de um evento de transição, verificar em qual dos modelos de utilização do tempo existentes se enquadra a nova rotina do utilizador. No caso de não enquadramento em nenhum dos existentes, deve proceder à criação de um novo, o comportamento implementado por omissão atualmente.

Em relação à deteção dos locais com significado para o utilizador seria interessante explorar uma aproximação de *clustering grid-based* (2.5) integrando com o sistema Geohash e avaliar os resultados obtidos em comparação com a solução implementada atualmente.

No que diz respeito a utilização geral da aplicação é fulcral estudar formas de diminuir o consumo de bateria sem prejudicar a qualidade dos algoritmos de extração de informação.

Tendo a aplicação um grau de maturidade a nível de interação que torne a sua utilização apelativa é muito relevante testa-lá com um grupo de utilizadores maior e mais diversificado que permita expor problemas não antecipáveis na modelação da rotina e na extração dos locais relevantes para o utilizador.

Bibliografia

[AAD+10] Gennady Andrienko, Natalia Andrienko, Urska Demsar, Doris Dransch, Jason Dykes, Sara Irina Fabrikant, Mikael Jern, Menno-Jan Kraak, Heidrun Schumann, e Christian Tominski. Space, time and visual analytics. *International Journal of Geographical Information Science*, 24(10):1577–1600, 2010.

[AAG03] Natalia V. Andrienko, Gennady L. Andrienko, e Peter Gatalsky. Exploratory spatio-temporal visualization: an analytical review. *Journal of Visual Languages and Computing*, 2003.

[BP00] Paramvir Bahl e Venkata N. Padmanabhan. Radar: an in-building rf-based user location and tracking system. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, pág. 775–784, 2000.

[con] Wikipedia contributors. Geohash. `http://en.wikipedia.org/wiki/Geohash`, Último Acesso : 21/06/2011.

[CQ11] Weiwei Cui e Huamin Qu. A survey on graph visualization. *Clinical Infectious Diseases*, 155(2), 2011.

[dB11] Samuel del Bello. Extração de informação de padrões pessoais de tempo e espaço. Tese de Mestrado, Universidade Nova de Lisboa, Faculdade de Ciências e Tecnologia, 2011.

[FGP08] Katayoun Farrahi e Daniel Gatica-Perez. What did you do today ? discovering daily routines from large-scale mobile data, 2008.

[HB01] Jeffrey Hightower e Gaetano Borriello. A survey and taxonomy of location systems for ubiquitous computing. Relatório técnico, University of Washington, Computer Science and Engineering, 2001.

[KK06] Ellegård Kajsa e Vrotsou Katerina. Capturing patterns of everyday life : presentation of the visualization method visual-timepacts. In *International Association for Time Use Research Annual Conference*. Danish National Institute of Social Research, 2006.

[KWSB05] J.H. Kang, W. Welbourne, B. Stewart, e G. Borriello. Extracting places from traces of locations. In *ACM SIGMOBILE Mobile Computing and Communications Review*, volume 9, pág. 58–68, 2005.

[Lam05] P. F. Lammertsma. Satellite navigation. Relatório técnico, Institute of Information and Computing Sciences, Utrecht University, 2005.

[MGP10] Raul Montoliu e Daniel Gatica-Perez. Discovering human places of interest from multimodal mobile phone data. In *Proceedings of the 9th International Conference on Mobile and Ubiquitous Multimedia*, MUM '10, pág. 12:1–12:10, New York, NY, USA, 2010. ACM.

[Mue11] Thomas Mueller. *H2 Database Engine*, 1.3.160 edition, 11 2011. http://www.h2database.com/html/main.html.

[Per11] Olivier Perriquet. Timemachine final report. Relatório técnico, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, 2011.

[PM09] Andrei Papliatseyeu e Oscar Mayora. Mobile habits: Inferring and predicting user activities with a location-aware smartphone. In Juan Corchado, Dante Tapia, e José Bravo, editores, *3rd Symposium of Ubiquitous Computing and Ambient Intelligence 2008*, volume 51 of *Advances in Soft Computing*. Springer Berlin / Heidelberg, 2009.

[Res] Hwaci Applied Software Research. Sqlite documentation. Último Acesso: 21/06/2011, http://www.sqlite.org/docs.html.

[RMI07] Jun Rekimoto, Takashi Miyaki, e Takaaki Ishizawa. Lifetag: Wifi-based continuous location logging for life pattern analysis. In *Proceedings of the 3rd international conference on Location-and context-awareness*, LoCA'07, pág. 35–49. Springer-Verlag, 2007.

[SlM08] Zeqian Shen e Kwan liu Ma. Mobivis: A visualization system for exploring mobile data. In *Asia Pacific Symposium on Information Visualisation*, pág. 175–182, 2008.

[Tuf90] Edward Tufte. *Envisioning Information*. Graphics Press, 1990.

[VW07] Fernanda B. Viégas e Martin Wattenberg. Artistic data visualization: beyond visual analytics. In *Proceedings of the 2nd international conference on Online communities and social computing*, OCSC'07, pág. 182–191, Berlin, Heidelberg, 2007. Springer-Verlag.

[WVH07] Martin Wattenberg, Fernanda B. Viégas, e Katherine Hollenbach. Visualizing activity on wikipedia with chromograms. In *Proceedings of the 11th IFIP TC 13 international conference on Human-computer interaction - Volume Part II*, INTERACT'07, pág. 272–287, Berlin, Heidelberg, 2007. Springer-Verlag.

[ZZF+04] Changqing Zhou, Changqing Zhou, Dan Frankowski, Pamela Ludford, Shashi Shekhar, e Loren"Terveen. Discovering personal gazetteers: An interactive clustering approach. *IN PROC. ACMGIS*, 2004:266–273, 2004.

[ZZXY10] Vincent W. Zheng, Yu Zheng, Xing Xie, e Qiang Yang. Collaborative location and activity recommendations with gps history data. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pág. 1029–1038. ACM, 2010.

7

Modelo Inquérito

Anexo que apresenta o modelo de inquérito realizado aos utilizadores.

Questionário Time Machine

O presente inquérito visa a recolha de informação sobre a experiência de utilização da aplicação móvel Time Machine. Todos os dados recolhidos são confidenciais e não serão utilizados para qualquer outra finalidade. A sua participação é importante na avaliação do trabalho efetuado, sendo uma contribuição de valor para o desenvolvimento da aplicação.

Informação Utilizador

Idade _____

Género Masculino ○ Feminino ○

Nível de Escolaridade

○ Doutoramento	○ Mestrado
○ Licenciatura	○ Ensino Secundário
○ 2º Ciclo do Ensino Básico	○ 3º Ciclo do Ensino Básico
○ 1º Ciclo do Ensino Básico	○ Autodidacta

Aptidão Tecnológica

Está familiarizado com as novas tecnologias ? Sim ○ Não ○

Que tipo de novas tecnologias utiliza ?

	Diariamente	Semanalmente	Mensalmente	Raramente	Nunca
Computador	○	○	○	○	○
Tablet	○	○	○	○	○
Telemóvel	○	○	○	○	○
Internet	○	○	○	○	○

Classifique a presença do telemóvel na sua vida.

Totalmente Ausente				Totalmente Presente
1	2	3	4	5
○	○	○	○	○

Estilo de Vida

Classifique a sua vida segundo o nível de atividade e rotina a que está sujeito(a).

	Menos				**Mais**
	1	2	3	4	5
Atividade	○	○	○	○	○
Rotina	○	○	○	○	○

	Sim	**Não**
Consegue definir claramente um local como sendo a sua Casa?	○	○
Trabalha num local fixo ?	○	○
Está sujeito(a) diariamente a horários regulares ?	○	○

Aplicação Móvel Time Machine (Geral)

Escolha <u>três (3)</u> adjetivos que melhor caracterizam a sua experiência de utilização da aplicação.

Interessante	Informativa	Simples	Desafiante	Imersiva
Boa	Divertida	Complexa	Motivante	Inovadora
Inconsequente	Impressionante	Irritante	Desagradável	Aborrecida
Frustrante	Confusa	Má	Péssima	Indiferente

Considera que a aplicação prejudica a utilização normal do telemóvel ?

Pouco				**Muito**
1	2	3	4	5
○	○	○	○	○

Classifique a identificação dos locais importantes para a sua vida feita pela aplicação (analise o ficheiro Kml que contém todos os locais identificados).

Incompleta				**Completa**
1	2	3	4	5
○	○	○	○	○

Classifique a noção de rotina que as diversas visualizações transmitem em conjunto.

Ineficaz				**Eficaz**
1	2	3	4	5
○	○	○	○	○

Aplicação Móvel Time Machine (Visualizações)

Responda apenas a visualizações que tenha utilizado. Deixe em branco caso contrário.

Visualização "Distância Diária Relativa" 

Considera que a visualização transmite as relações entre as distâncias percorridas nos diferentes dias ?

Não Transmite				**Transmite Totalmente**
1	2	3	4	5
○	○	○	○	○

Considera que a visualização transmite a quantidade de distância percorrida em cada dia ?

Não Transmite				**Transmite Totalmente**
1	2	3	4	5
○	○	○	○	○

Avalie a fluidez da deslocação ao longo dos dias (manipulação tátil do ecrã).

Nada Fluída				**Totalmente Fluída**
1	2	3	4	5
○	○	○	○	○

Escolha <u>três (3)</u> adjetivos que melhor caracterizam a visualização.

Interessante	Informativa	Simples	Desafiante	Imersiva
Boa	Divertida	Complexa	Motivante	Inovadora
Inconsequente	Impressionante	Irritante	Desagradável	Aborrecida
Frustrante	Confusa	Má	Péssima	Indiferente

Visualização "Figura Geométrica" 

Considera que a visualização transmite o nível de atividade a que se sente sujeito ao longo do dia ?

Não Transmite				**Transmite Totalmente**
1	2	3	4	5
○	○	○	○	○

Considera um desafio fazer com que o quadrado se deforme mais?

Nada Desafiante				**Totalmente Desafiante**
1	2	3	4	5
○	○	○	○	○

Escolha <u>três (3)</u> adjetivos que melhor caracterizam a visualização.

Interessante	Informativa	Simples	Desafiante	Imersiva
Boa	Divertida	Complexa	Motivante	Inovadora
Inconsequente	Impressionante	Irritante	Desagradável	Aborrecida
Frustrante	Confusa	Má	Péssima	Indiferente

Visualização "Qual é a cor deste Local?"

De acordo com o código de cor, em que amarelo significa um local muito conhecido, e vermelho totalmente desconhecido, considera que a visualização transmite o conhecimento do local onde se encontra a cada momento ?

Não Transmite				**Transmite Totalmente**
1	2	3	4	5
○	○	○	○	○

De que forma, em termos de rapidez, considera que a visualização reage ao local onde se encontra ?

Lentamente				**Rapidamente**
1	2	3	4	5
○	○	○	○	○

Escolha <u>três (3)</u> adjetivos que melhor caracterizam a visualização.

Interessante	Informativa	Simples	Desafiante	Imersiva
Boa	Divertida	Complexa	Motivante	Inovadora
Inconsequente	Impressionante	Irritante	Desagradável	Aborrecida
Frustrante	Confusa	Má	Péssima	Indiferente

Visualização Quadrados de familiaridade 

Sabendo que claridade representa familiariedade e escuridão desconhecimento, considera que a visualização transmite este conceito de acordo com o dia e o local em que se encontra?

Não Transmite				**Transmite Totalmente**
1	2	3	4	5
○	○	○	○	○

De que forma, em termos de rapidez, considera que a visualização reage ao local onde se encontra ?

Lentamente				**Rapidamente**
1	2	3	4	5
○	○	○	○	○

Escolha <u>três (3)</u> adjetivos que melhor caracterizam a visualização.

Interessante	Informativa	Simples	Desafiante	Imersiva
Boa	Divertida	Complexa	Motivante	Inovadora
Inconsequente	Impressionante	Irritante	Desagradável	Aborrecida
Frustrante	Confusa	Má	Péssima	Indiferente

Visualização Pequenos Retângulos de Tempo 

Sabendo que uma cor mais escura representa um afastamento do comportamento considerado como rotina em termos de atividade, considera que a visualização transmite os padrões horários em que tem uma atividade diferente do habitual ?

Não Transmite				**Transmite Totalmente**
1	2	3	4	5
○	○	○	○	○

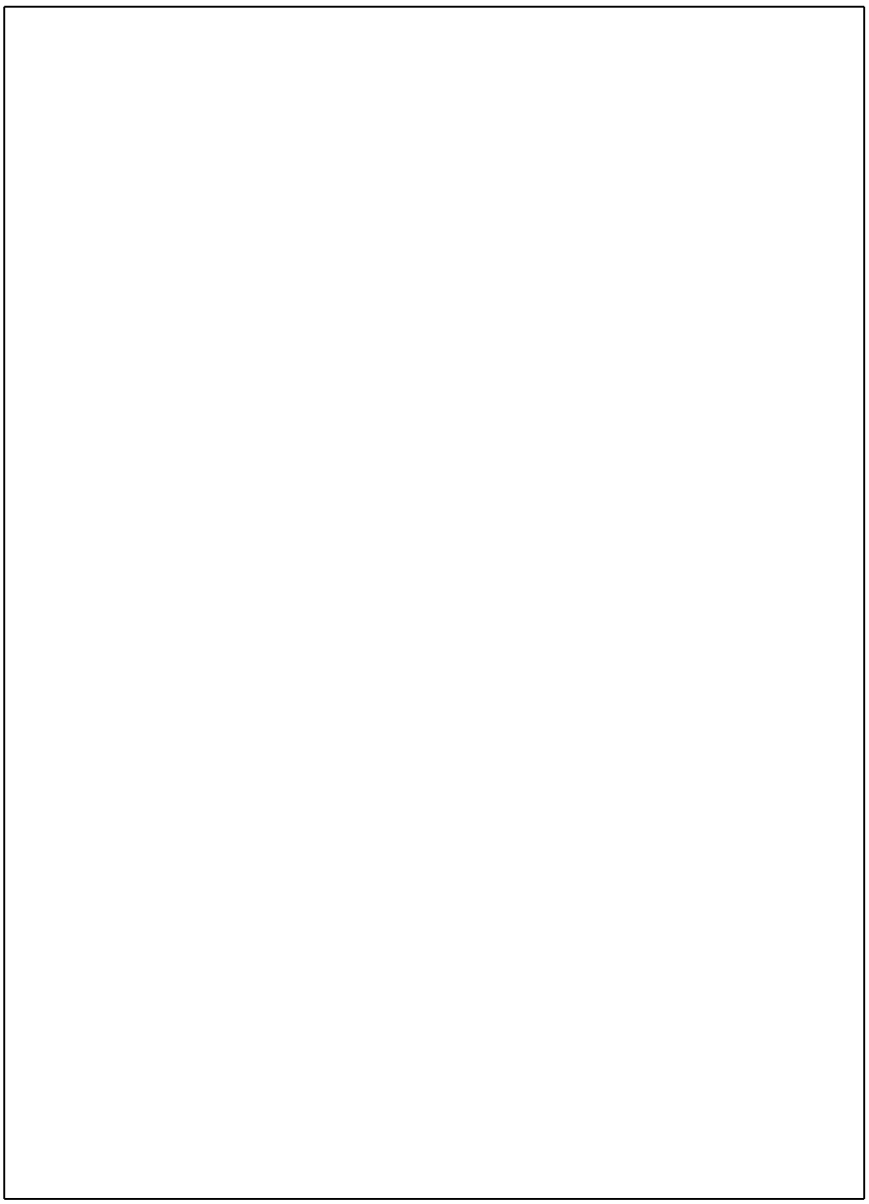
Olhando para as horas futuras (para lá da linha divisória) consegue estabelecer uma relação entre o que prevê que venha a acontecer e a sugestão transmitida pela visualização?

Raramente				**Frequentemente**
1	2	3	4	5
○	○	○	○	○

Escolha <u>três (3)</u> adjetivos que melhor caracterizam a visualização.

Interessante	Informativa	Simples	Desafiante	Imersiva
Boa	Divertida	Complexa	Motivante	Inovadora
Inconsequente	Impressionante	Irritante	Desagradável	Aborrecida
Frustrante	Confusa	Má	Péssima	Indiferente

Sugestões e Comentários



8

Resultado dos Inquéritos

Anexo contendo os resultados dos inquéritos realizados. A apresentação segue a mesma sequência das questões no inquérito.

Faixa Etária	Utilizadores
Menos de 10	0
10 a 19	1
20 a 29	4
30 a 39 10	0
40 a 49	1
50 a 59	1
Mais de 60	0

Tabela 8.1: Informação utilizador: idade

Género	Utilizadores
Masculino	5
Feminino	2

Tabela 8.2: Informação utilizador: género

Escolaridade	**Utilizadores**
Douturamento	0
Mestrado	2
Licenciatura	3
Ensino Secundário	1
3ºCiclo	0
2ºCiclo	1
1ºCiclo	0

Tabela 8.3: Informação utilizador: escolaridade

Familiaridade com Novas Tecnologias	**Utilizadores**
Sim	7
Não	0

Tabela 8.4: Informação utilizador: familiaridade com novas tecnologias

Tecnologias Utilizadas	**Diariamente**	**Semanalmente**	**Mensalmente**	**Raramente**	**Nunca**
Computador	7	0	0	0	0
Tablet	1	1	1	3	1
Telemóvel	7	0	0	0	0
Internet	1	0	0	0	0

Tabela 8.5: Informação utilizador: tecnologias utilizadas

Presença Telemóvel	Utilizadores
1	0
2	0
3	2
4	0
5	5

Tabela 8.6: Informação utilizador: presença telemóvel

Atividade	Utilizadores
1	0
2	1
3	2
4	4
5	0

Tabela 8.7: Estilo de vida: atividade

Rotina	Utilizadores
1	0
2	1
3	1
4	2
5	3

Tabela 8.8: Estilo de vida: rotina

Definir Local da sua Casa	Utilizadores
Sim	5
Não	2

Tabela 8.9: Estilo de vida: definir local da sua casa

Local de Trabalho Fixo	Utilizadores
Sim	5
Não	2

Tabela 8.10: Estilo de vida: local de trabalho fixo

Horários Regulares	Utilizadores
Sim	6
Não	1

Tabela 8.11: Estilo de vida: horários regulares

Experiência de Utilização	Utilizadores
Interessante	7
Informativa	3
Simples	0
Desafiante	2
Imersiva	0
Boa	0
Divertida	0
Complexa	3
Motivante	2
Inovadora	4
Inconsequente	0
Impressionante	0
Irritante	0
Desagradável	0
Aborrecida	0
Frustrante	0
Confusa	0
Má	0
Péssima	0
Indiferente	0

Tabela 8.12: Aplicação móvel Time Machine (geral): experiência de utilização

Impacto Utilização do Telemóvel	Utilizadores
1	0
2	0
3	2
4	3
5	2

Tabela 8.13: Aplicação móvel Time Machine (geral): impacto utilização do telemóvel

Extração de Locais com Significado	Utilizadores
1	0
2	0
3	0
4	3
5	4

Tabela 8.14: Aplicação móvel Time Machine (geral): extração de locais com significado

Transmissão Noção de Rotina	Utilizadores
1	0
2	0
3	3
4	4
5	0

Tabela 8.15: Aplicação móvel Time Machine (geral): transmissão noção de rotina

Experiência de Utilização	Utilizadores
Interessante	3
Informativa	6
Simples	3
Desafiante	0
Imersiva	0
Boa	2
Divertida	0
Complexa	1
Motivante	0
Inovadora	1
Inconsequente	1
Impressionante	0
Irritante	0
Desagradável	0
Aborrecida	1
Frustrante	0
Confusa	2
Má	0
Péssima	0
Indiferente	1

Tabela 8.16: Visualização Distância Diária Relativa: experiência de utilização

Relação entre Distâncias Diárias	Utilizadores
1	0
2	0
3	1
4	4
5	2

Tabela 8.17: Visualização Distância Diária Relativa: relação entre distâncias diárias

Quantidade Distância	Utilizadores
1	0
2	0
3	2
4	3
5	2

Tabela 8.18: Visualização Distância Diária Relativa: quantidade distância

Fluidez	Utilizadores
1	0
2	0
3	1
4	4
5	2

Tabela 8.19: Visualização Distância Diária Relativa: fluidez

Experiência de Utilização	Utilizadores
Interessante	1
Informativa	2
Simples	1
Desafiante	5
Imersiva	0
Boa	2
Divertida	4
Complexa	0
Motivante	0
Inovadora	6
Inconsequente	0
Impressionante	0
Irritante	0
Desagradável	0
Aborrecida	0
Frustrante	0
Confusa	0
Má	0
Péssima	0
Indiferente	0

Tabela 8.20: Visualização Forma Geométrica: experiência de utilização

Transmissão Atividade	Utilizadores
1	0
2	0
3	0
4	4
5	3

Tabela 8.21: Visualização Forma Geométrica: transmissão atividade

Desafio	Utilizadores
1	0
2	0
3	0
4	3
5	4

Tabela 8.22: Visualização Forma Geométrica: desafio

Experiência de Utilização	Utilizadores
Interessante	1
Informativa	4
Simples	5
Desafiante	0
Imersiva	0
Boa	2
Divertida	2
Complexa	0
Motivante	0
Inovadora	2
Inconsequente	1
Impressionante	0
Irritante	0
Desagradável	0
Aborrecida	0
Frustrante	0
Confusa	0
Má	0
Péssima	0
Indiferente	1

Tabela 8.23: Visualização Qual a cor do Local?: experiência de utilização

Transmissão Conhecimento	Utilizadores
1	0
2	0
3	1
4	4
5	1

Tabela 8.24: Visualização Qual a cor do Local?: transmissão conhecimento

Rapidez	Utilizadores
1	0
2	0
3	3
4	2
5	1

Tabela 8.25: Visualização Qual a cor do Local?: rapidez

Experiência de Utilização	Utilizadores
Interessante	3
Informativa	3
Simples	1
Desafiante	0
Imersiva	1
Boa	1
Divertida	0
Complexa	1
Motivante	0
Inovadora	3
Inconsequente	0
Impressionante	0
Irritante	0
Desagradável	0
Aborrecida	1
Frustrante	0
Confusa	1
Má	0
Péssima	0
Indiferente	0

Tabela 8.26: Visualização Quadrados de Familiaridade: experiência de utilização

Transmissão rotina na utilização do local	Utilizadores
1	0
2	0
3	2
4	2
5	1

Tabela 8.27: Visualização Quadrados de Familiaridade: transmissão rotina na utilização do local

Rapidez	Utilizadores
1	0
2	0
3	2
4	1
5	2

Tabela 8.28: Visualização Quadrados de Familiaridade: rapidez

Experiência de Utilização	Utilizadores
Interessante	5
Informativa	4
Simples	2
Desafiante	1
Imersiva	0
Boa	0
Divertida	0
Complexa	0
Motivante	0
Inovadora	3
Inconsequente	0
Impressionante	0
Irritante	0
Desagradável	0
Aborrecida	0
Frustrante	0
Confusa	0
Má	0
Péssima	0
Indiferente	0

Tabela 8.29: Visualização Pequenos Retângulos de Tempo: experiência de utilização

Transmissão rotina horária de atividade	Utilizadores
1	0
2	0
3	1
4	2
5	2

Tabela 8.30: Visualização Pequenos Retângulos de Tempo: transmissão rotina horária de atividade

Previsão	Utilizadores
1	0
2	0
3	1
4	3
5	1

Tabela 8.31: Visualização Pequenos Retângulos de Tempo: previsão